



**DOCUMENT 169-04
(SUPPLEMENT)**

**DATA REDUCTION AND COMPUTER
GROUP**

**COMMON RANGE ARCHITECTURE OBJECT MODEL
APPROVAL PROCESS INVESTIGATION**

**WHITE SANDS MISSILE RANGE
REAGAN TEST SITE
YUMA PROVING GROUND
DUGWAY PROVING GROUND
ABERDEEN TEST CENTER
NATIONAL TRAINING CENTER
ELECTRONIC PROVING GROUND**

**NAVAL AIR WARFARE CENTER WEAPONS DIVISION
NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION
NAVAL UNDERSEA WARFARE CENTER DIVISION, NEWPORT
PACIFIC MISSILE RANGE FACILITY
NAVAL UNDERSEA WARFARE CENTER DIVISION, KEYPORT**

**30TH SPACE WING
45TH SPACE WING
AIR FORCE FLIGHT TEST CENTER
AIR ARMAMENT CENTER
AIR WARFARE CENTER
ARNOLD ENGINEERING DEVELOPMENT CENTER
BARRY M. GOLDWATER RANGE
UTAH TEST AND TRAINING RANGE**

NATIONAL NUCLEAR SECURITY ADMINISTRATION (NEVADA)

**DISTRIBUTION A: APPROVED FOR PUBLIC RELEASE
DISTRIBUTION IS UNLIMITED**

This page intentionally left blank.

**DOCUMENT 169-04
(SUPPLEMENT)**

**COMMON RANGE ARCHITECTURE OBJECT MODEL
APPROVAL PROCESS INVESTIGATION**

NOVEMBER 2004

Prepared by

**DATA REDUCTION AND COMPUTER GROUP (DR&CG)
RANGE COMMANDERS COUNCIL**

Published by

**Secretariat
Range Commanders Council
U.S. Army White Sands Missile Range,
New Mexico 88002-5110**

THIS DOCUMENT IS AVAILABLE ON THE
RANGE COMMANDERS WEBSITE AT
<http://jcs.mil/RCC>

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF APPENDIX FIGURES	iv
PREFACE.....	v
ACRONYMS.....	vii
CHAPTER 1: EXECUTIVE SUMMARY.....	1-1
CHAPTER 2: DATA REDUCTION TASK DR-31 BREAKDOWN.....	2-1
2.1 The Need For An Object Model (OM) Standardization Process	2-1
2.2 DR-31 Effort: Task 1 and Task 2 Defined.....	2-1
CHAPTER 3: TASK 2: DEVELOP PROCESS TO STORE, REVIEW, MODIFY, AND MANAGE OBJECT MODELS.....	3-1
3.1 Background and Approach	3-1
3.2 Technology Background.....	3-1
3.3 Deliverables For Task 2	3-13
3.4 Prototype On-Line OM Submission Process	3-15
APPENDIX A: METADATA FIELDS.....	A-1
APPENDIX B: FOUNDATION INITIATIVE 2010 USE CASE TEMPLATES.....	B-1
APPENDIX C: FI 2010 TENA USE CASE INSTRUCTIONS.....	C-1
1.1 Introduction.....	C-1
1.2 Use Case Template Instructions.....	C-2
APPENDIX D: EXAMPLE RADAR USE CASE.....	D-1
Bibliography	

LIST OF FIGURES

Figure 3-1.	Use Case representation in UML.....	3-5
Figure 3-2.	Actor representation in UML.....	3-5
Figure 3-3.	Class diagram for the player object.....	3-6
Figure 3-4.	Inheritance.....	3-7
Figure 3-5.	Abstraction.....	3-8
Figure 3-6.	Composition.....	3-8
Figure 3-7.	Sequence diagram.....	3-9
Figure 3-8.	Deployment diagram.....	3-10
Figure 3-9.	The Developmental Test Command (DTC) integration level hierarchy.....	3-11
Figure 3-10.	TENA metamodel.....	3-12
Figure 3-11.	HLA metamodel based on the Object Model template.....	3-13
Figure 3-12.	Website home page.....	3-16
Figure 3-13.	Website login.....	3-16
Figure 3-14.	Object Model submission.....	3-17
Figure 3-15.	Object Model submission - completed.....	3-18
Figure 3-16.	DR&CG Chair login.....	3-18
Figure 3-17.	Review team created by Chair.....	3-19
Figure 3-18.	Review team created by Chair - Completed.....	3-19
Figure 3-19.	Review team login.....	3-20
Figure 3-20.	Review team comment.....	3-20
Figure 3-21.	Review team comment – completed.....	3-21
Figure 3-22.	Working group creation by chair.....	3-22
Figure 3-23.	Working group creation by Chair - completed.....	3-22
Figure 3-24.	Working group login.....	3-23
Figure 3-25.	Working group choices.....	3-23
Figure 3-26.	Working group comments.....	3-24
Figure 3-27.	Working group acceptance of topic.....	3-25
Figure 3-28.	Review Team monitoring working group status.....	3-25
Figure 3-29.	DR&CG Chair acceptance.....	3-26

LIST OF APPENDIX FIGURES

Figure B-1.	TENA Use Case basic information.....	B-1
Figure B-2.	TENA Applications Worksheet.....	B-2
Figure B-3.	Events Worksheet.....	B-2
Figure B-4.	Information Flow Worksheet.....	B-3
Figure B-5.	Object Model Worksheet.....	B-3
Figure B-6.	Other Information Worksheet.....	B-4
Figure C-1.	The Requirements Analysis Process for the TENA Middleware Prototype.....	C-1
Figure C-2.	Connectivity diagram example.....	C-4
Figure D-1.	Example radar Use Case (basic information).....	D-1
Figure D-2.	Example radar Use Case (TENA applications).....	D-2
Figure D-3.	Example radar Use Case (basic course of events).....	D-2
Figure D-4.	Example radar Use Case (information flows).....	D-2
Figure D-5.	Example radar Use Case (Object Model information).....	D-3
Figure D-6.	Example radar Use Case (other relevant information).....	D-4

PREFACE

The Range Commanders Council (RCC) Data Reduction and Computer Group (DR&CG) sponsored the development and publication of this document. This document represents the release of Task 2 of the DR&CG study effort DR-31, “Common Range Architecture Approval Process Investigation.” The DR&CG developed this document as a supplement to Document 169-04 to provide the reader with detailed guidelines for developing Object Models (OM). The goal is for consistency to be achieved in developing OM standards throughout the Department of Defense (DoD).

The primary contributors to this report are shown below.

Author: Mr. Kurt Lessmann
Spring City Solutions, Inc.
106 Twin Island Circle
Madison, AL 35758
E-mail: klessmann@springcitysolutions.com

Mr. David Browning
Data Reduction And Computer Group (DR&CG), Associate Member
Representing: Redstone Technical Test Center (CSTE-DTC-RT-F-FL)
Redstone Arsenal, AL 35898-8052
E-Mail: dbrowning@rttc.army.mil

This page intentionally left blank.

ACRONYMS

ABBREVIATIONS

TERMS

C++	An Object Oriented computer-programming language based on the C language
C	A powerful and flexible computer programming language that can be used for a variety of applications, from business to engineering programs.
COTS	Commercial off the shelf
CRAC	Common Range Architecture Committee
CTEIP	Central Test and Evaluation Investment Program
DAT	Development Advisory Team
DR&CG	Data Reduction and Computer Group
DoD	Department of Defense
DR-31	Identifies Data Reduction and Computer Group Task Number 31
DTC	Developmental Test Command (DTC)
FI2010	Foundation Initiative 2010
FIPMO	Foundation Initiative 2010 Project Management Office
ILH	Integration Level Hierarchy
JPEG	Joint Photographic Experts Group
OM	Object Model
OMG	Object Management Group
OO	Object Oriented
OOP	Object Oriented Programming
OS	Operating System
POC	Point of Contact
RCC	Range Commanders Council
SDO	Stateful distributed object
TENA	Test and Training Enabling Architecture
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

This page intentionally left blank.

CHAPTER 1

EXECUTIVE SUMMARY

The charter for the Range Commanders Council (RCC) Data Reduction and Computer Group (DR&CG), Common Range Architecture Committee (CRAC), includes the evaluation of proposed RCC architectural standards as well as the configuration management and distribution of candidate and accepted standards.

The Central Test and Evaluation Investment Program (CTEIP) Foundation Initiative 2010 (FI2010) project has developed the Test and Training Enabling Architecture (TENA) to support test and training range interoperability. As part of the TENA objective, the FI2010 project will be offering proposed architectural standards to the RCC for ratification and management. The first offering from the project will be the common Object Models (OM) being produced and utilized within the TENA architecture. A pathfinder project was established to articulate the issues, provide a process for Object Model standardization, and prepare a guideline for standardization. This pathfinder project is known as the RCC task DR-31 effort, “Common Range Architecture Object Model Approval Process Investigation.”

This document defines the detailed process that the DR&CG will use to store, review, modify, and manage the Object Models as they progress through the standardization process defined in Task 1. This document addresses Task 2, which was established to provide the detailed data management definitions and supporting technologies necessary to properly manage the submission, review and maintenance of proposed Object Models.

This documents presents the findings of Spring City Solutions, Inc in support of DR-31 Task 2 with the following subtasks:

- a. Develop draft formats that should be used for Object Model submission to the RCC.
- b. Develop draft formats for the distributed and archived Object Models.
- c. Develop draft process by which Object Models are submitted, updated, distributed, reviewed, modified, and archived.

The deliverable information for subtask a and subtask b is found: **Chapter 3**, [paragraph 3.2](#), and [paragraph 3.3](#). Subtask c deliverable information is found in Chapter 3, [paragraph 3.4](#).

This page intentionally left blank.

CHAPTER 2

DATA REDUCTION TASK DR-31 BREAKDOWN

2.1 The Need For An Object Model (OM) Standardization Process

There are currently many activities that strive to enable interoperability between ranges and range resources. Therefore, a significant portion of these activities support the standardization of the data passed between the ranges. In addition, the architectures developed to support range interoperability, such as the Test and Training Enabling Architecture (TENA), have adopted an Object Oriented (OO) approach. When OO based software is used in conjunction with data standardization, a notion of Object Model (OM) is presented. An OM is the interface to a given system that describes its data and functional capabilities. In other words, it's the "contract" that must be enforced to support interoperability. The RCC task DR-31, Common Range Architecture Object Model Approval Process Investigation, was initiated to address concerns regarding the process required to standardize proposed Object Models.

2.2 DR-31 Effort: Task 1 and Task 2 Defined

The DR-31 effort was established to support two main tasks:

- a. Task 1 - Develop the initial high-level notional process by which the RCC in general, and the DR&CG in particular, should standardize Object Models.
- b. Task 2 - Develop the high-level notional process that the RCC could store, review, modify, and manage the Object Models as they progress through the standardization process defined in Task 1.

2.2.1 Task 1 Deliverables. For Task 1, the primary process deliverables are in shown in Chapter 3 of Document 169-04, Common Range Architecture Object Model Approval Process Investigation. The primary deliverables include guidelines as to when a candidate OM should be reviewed, a draft process for review by Object Model experts, and a draft process for revision and final approval of candidate Object Models. Additional deliverables include tutorial and training materials on software architectures and Object Oriented concepts.

2.2.2 Task 2 Deliverables. As Task 1 was established to define an over-arching OM submission process, Task 2 was established to provide the detailed data management definitions and supporting technologies necessary to properly manage the submission, review, and maintenance of proposed OMs. The deliverables as set forth by DR-31 Task 2 are contained in the remaining portions of this document, including:

- a. Draft formats that should be used for OM submission to the RCC groups
- b. Draft formats for the distributed and archived Object Models
- c. Draft process by which Object Models are submitted, updated, distributed, reviewed, modified and archived

This page intentionally left blank.

CHAPTER 3

TASK 2: DEVELOP PROCESS TO STORE, REVIEW, MODIFY, AND MANAGE OBJECT MODELS

3.1 Background and Approach

The deliverable items of the DR-31 effort were outlined in [paragraph 2.2.2](#) above. This chapter documents the findings of Spring City Solutions, Inc., in support of DR-31, Task 2. The following three sections provide the information necessary to fulfill the deliverables of Task 2.

- a. Paragraph 3.2 defines terms and technologies required to discuss the requirements and proposed solution.
- b. [Paragraph 3.3](#) presents the deliverables for Task 2.
- c. [Paragraph 3.4](#) presents an example implementation of the process as defined in the DR-31 Task 1 and the formats/process as defined in DR-31 Task 2.

3.2 Technology Background

As with any technology-based area of interest, there often is an underlying level of confusion on the terminology used to define the technology. Therefore, several terms will be defined in the following sections to present a baseline of definitions. This information baseline will enable a common understanding of terms, and thereby provide a common understanding of the technology.

3.2.1 Object Oriented Programming (OOP). Fundamentally, the use of OOP combines both the data and the functions that operate on that data into a single unit, called an object. In many cases the term behavior is interchangeable with the concept of object functions that operate on the data. As a general rule, objects are defined when the software program must represent both data and behavior of something in the real world.

An Object Oriented (OO) program is basically a collection of objects that interact with each other to provide a function or capability. To illustrate this concept, let's consider a program that represents a baseball team. For the program to represent a real baseball team, at a minimum there needs to be representations of the players on the baseball team. These representations are called *objects*, that have been named "Player," and each Player object has data and behavior.

The data stored within an object are referred to as *attributes* in OOP terminology. In our Player objects, they would need attributes that represent the real-life player attributes. Potential attributes defined for the Player object might be position, battingAverage, onBasePercentage, number, and other characteristics of the actual player.

The behavior of an object is the functionality that the object represents. In OOP terminology, the object behavior is represented by "methods." A method may be *invoked*, or executed, by other objects by providing the data that the method needs to execute the method. This data being passed to the method are called *parameters*. If the method passes data back after executing the method, the type of data being returned must be specified and is called the *return*

type. In the Player example, a method would be *StealBase*, the parameter would be *baseNumber*, and the return type would be *Boolean* for true or false.

Another OOP term that needs to be addressed is *class*. A class is a template, or blueprint, of an object. A good way to describe the concept is by using the construction of a building as an analogy. Before a general contractor starts construction on a building, he needs a set of blueprints to construct the building so that it meets the dimensions and behavior of the owners. Likewise, the software developer needs a set of plans before an object can be created in a program. The class definitions come together to form the required set of plans. Therefore, a class is to a software developer as a blueprint is to a general contractor.

Inheritance is an OOP term that represents the ability of a class to reuse or inherit the attributes and methods of another class. This inheritance allows a developer to create a brand new type of class that is more specialized than the “base” class. For example, since all baseball players have the common characteristic of being human; we could therefore create a *super class* called “Human” with attributes like age, weight, sex and methods like run, walk, and sit. The *sub-class* Player could inherit from Human and thereby contain all the attributes and methods found in the Human class. The Player sub-class would have all the attributes needed to characterize a particular human as a baseball player. This type of relationship is an “IS-A” relationship (the Player is a Human). An “IS-A” relationship, or inheritance, signifies a very tight coupling meaning that the player MUST BE a Human.

Abstraction is a very powerful concept that allows additional sub-classes to be added quite easily. For instance, say you wanted to add “Coach” class as a sub-class to the super class Human. The Coach class would inherit all the elements of Human, but the Coach class would have additional attributes and methods that the Human and Player classes do not have. For example the Coach class could have a method called “sendStealBaseSignal” with a return type of void.

Interestingly, the class designer now must decide if the class Coach should be a sub-class to Human or a sub-class to Player, because both could be true. This example situation is raised to emphasize two points:

- a. Abstraction is multi-level.
- b. Although a person grasps OO concepts, he may or may not be able to design a good OO system

Composition refers to one object containing additional object(s). For example, a Player object may have a Uniform, Glove, and Bat objects. This type of relationship is called a “HAS-A” relationship because the Player HAS-A Glove, Bat and Uniform. But the Player may exist without the composed objects Glove, Bat, and Uniform.

While it is beyond the scope of this document to fully explain OOP terminology and skills, the previous sections provide the reader with enough background material to understand the necessary concepts.

3.2.2 Object Model definition. While the previous section provided the basics of OOP, the term “Object Model” was never mentioned. The term Object Model is actually somewhat of a misnomer that has been used for several years within the modeling and simulation (M&S) community. Unfortunately, it has been adopted as a pseudo-standard definition within the community. To adopt OOP terminology, an OM is the class definition(s) of a particular software program. The OM defines the data and behavior of a particular object in question and thereby making it a “contract” to allow communication with an external system.

The term “object,” as mentioned before, is defined in Object Oriented terminology as an instance of a class. Before an instance of the class in a software application can be created, the software developer needs to understand how the class is defined. The developer takes the definition of class and creates an object based on that class. The newly defined object then supports the data and behavior as defined by the class definition.

The term “model,” can have different meanings to different people. If several people are asked what the term “model” means to them, the responses may be like the following:

- a. A plastic toy that is put together with plastic glue.
- b. A 3-Dimensional synthetic representation of a live entity that is used in real-time graphics.
- c. A simple software algorithm.

Note: In OOP constructs, a model is used to designate a diagram, template, or layout.

In the above paragraphs, the term “class” was used to refer to a template, or definition, of the object to be instantiated (i.e. created) and the term “model” was used to refer to a diagram. With OO terminology, a more accurate term for “Object Model,” should really be “class diagram.” For the remainder of this document, the term “Object Model” should be thought of as a class diagram. The following paragraphs describe standard methodologies and techniques for defining and presenting OMs.

3.2.3 Representing Object Oriented systems. When the author of a C++ textbook from 10 years ago presented new OOP definitions and techniques, he often used C++ syntax to represent things like classes, methods, and attributes. The new user found the syntax difficult to use because the author was using C++ before the user knew C++. To counteract this difficulty, some authors developed a custom set of graphical “boxes” to represent classes and a set of arrows to represent relationships between the classes.

As OOP has become the accepted software design and development paradigm, several organizations have been working very diligently to formalize the process to describe, design, and develop software systems. For example, the Object Management Group (OMG) has been recognized as the world-leading organization that manages standards for interoperability between applications. From the OMG website, the OMG describes itself as “... an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Our membership includes virtually every large company in

the computer industry, and hundreds of smaller ones. Most of the companies that shape enterprise and Internet computing today are represented on our Board of Directors.”

One of the technologies the OMG has standardized is called the Unified Modeling Language (UML). The UML is a defined process that allows software developers and system architects to visualize, specify, construct, and document software systems. The UML is very well defined and documented with a formal specification of the “language,” or process. This powerful concept enables a standardized process for developing:

- a. Diagrams that describe Object Oriented (OO) information such as class and object diagrams. These diagrams illustrate the static structure and relationship between objects in a system.
- b. Interaction and activity diagrams to convey the dynamic behavior of objects in the system. The diagrams may also convey messages between the objects based on certain events or happenings.
- c. Use Case diagrams to ensure user/customer requirements are well documented and well understood by the system development team.
- d. Deployment diagrams to present the use of the software in a particular environment.

While the OMG doesn’t produce UML modeling tools, it does produce, standardize, and manage the UML specifications. These specifications are supported by Commercial Off The Shelf (COTS) UML tool developers like IBM Rational, Borland, No Magic, and Objectteering. Due to the varying degrees of adoption and implementation of the UML standards, data interchange problems sometimes exist between UML modeling tools.

Because there is now a technology called UML, the question arises as to how the UML technology should be used for standard OM submissions and definitions. UML uses a notation of *Views* and *Diagrams* to represent the system being modeled. Because there are several different UML views and many different UML diagrams, the following paragraphs address how several proposed UML diagrams should be used to not only define the proposed OM, but also to present the information regarding the definition of the OM.

3.2.4 UML diagrams. The features inherent in UML make it very powerful by allowing customers, managers, and software developers to collaborate on a design and document all aspects of the software system in a standardized way. UML diagrams have been conceived to enable information flow by presenting views in class, object, Use Case, state machine, sequence, communication, timing, interaction overview, activity, component and deployment diagrams. In the following sections, this report will focus on the following four critical UML diagrams needed to ensure that OMs are generated and presented in a standard acceptable definition.

- a. Use Case diagrams - Provides a graphical representation of the functionality needed.
- b. Class diagrams - Provides a graphical representation of the OM.
- c. Sequence diagrams - Provides an understanding of the sequence of events and the data flow between objects.
- d. Deployment diagrams - Provides a graphical representation of the physical configuration of the hardware.

3.2.4.1 Use Case diagrams. A Use Case diagram can be thought of as a package of information, typically presented in multiple diagrams, which describes what a system does, or should do, to benefit the stakeholder. The stakeholder can be thought of as the customer, the user, or the benefactor, of the software system being developed. The Use Case diagrams focus the overwhelming amount of reference data into a refined, concise information package that represents the intended functionality of the software system. This Use Case information is then utilized for discussions between the stakeholder, the software developer, and the system architect to define the requirements of the desired software system so that all understand and agree.

A Use Case model defined in UML notation is described by a number of Use Case diagrams. The Use Case diagrams contain model elements for the system, the actors, and the Use Cases and also depict relationships between the elements. The potentially confusing aspect here is that a Use Case defining a system will probably contain several “smaller” Use Cases. In UML, a Use Case is defined using an ellipsoid (see Figure 3-1) and contains a title that defines the activity/requirement of the Use Case. For instance a Use Case may be called “Batting Practice” that is a component of a “Baseball” Use Case.



Figure 3-1. Use Case representation in UML.

An “actor” is defined in UML as a “role” or a classifier in the Use Case. The name of the actor portrays the role that the actor represents. Each actor must have some role, or association, with at least one Use Case. If there is an actor without an association to a Use Case, then that actor is no longer needed. An actor always initiates Use Cases and Use Cases always provide value to the actor. The “stickman” diagram in Figure 3-2 represents an actor.

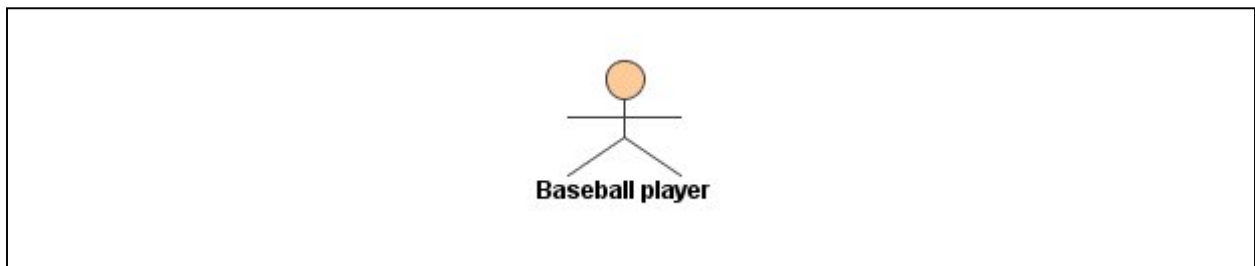


Figure 3-2. Actor representation in UML.

A description of the Use Case is provided in text, and is a simple explanation regarding:

- a. The objective of the Use Case.
- b. How the Use Case and the actors interact.
- c. How the Use Case is initiated.
- d. Message traffic between the actor(s) and the Use Case.
- e. Value to the actor.
- f. The exit criteria of the Use Case.

3.2.4.2 Class diagrams. As the name implies, UML is a modeling language complete with rules, notations, and diagrams. UML provides the user a methodology to represent the design of the system and also to design a software system. As mentioned previously, an Object Model is equal to that of a Class Diagram in UML notation. Unfortunately, presenting the entire UML class diagram notation tutorial is beyond the scope of this effort. However, the following sections will provide the requirements for understanding UML class diagram notation based on the Player example.

In UML, a class diagram is basically a rectangle with three horizontal lines separating the rectangle into three areas. Each area in the rectangle represents the three distinct parts of a class: the class name, attributes, and methods. The components of Figure 3-3 are as follows:

- a. Class Name - Top section contains the class name, “Player”
- b. Attributes - The second section contains the attributes and their associated data types such as attribute “position” and data type “short.”
- c. Methods - The third section contains the methods to include the arguments and return types such as method name “stealBase”, with argument “baseNumber”, data type “short”, and type of return method “Boolean”.

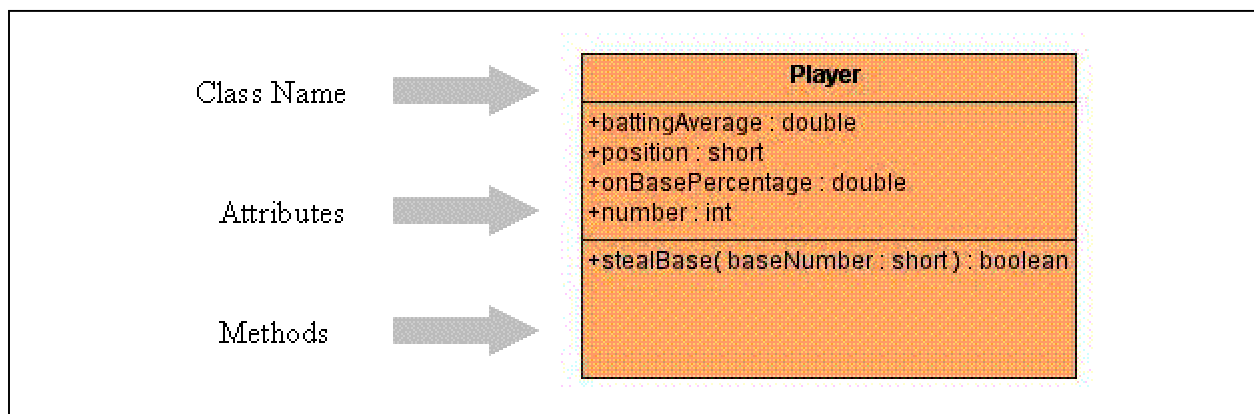


Figure 3-3. Class diagram for the Player object.

To define inheritance, a line is drawn from the sub-class to a super class with an open triangle towards the super class. Figure 3-4 represents inheritance.

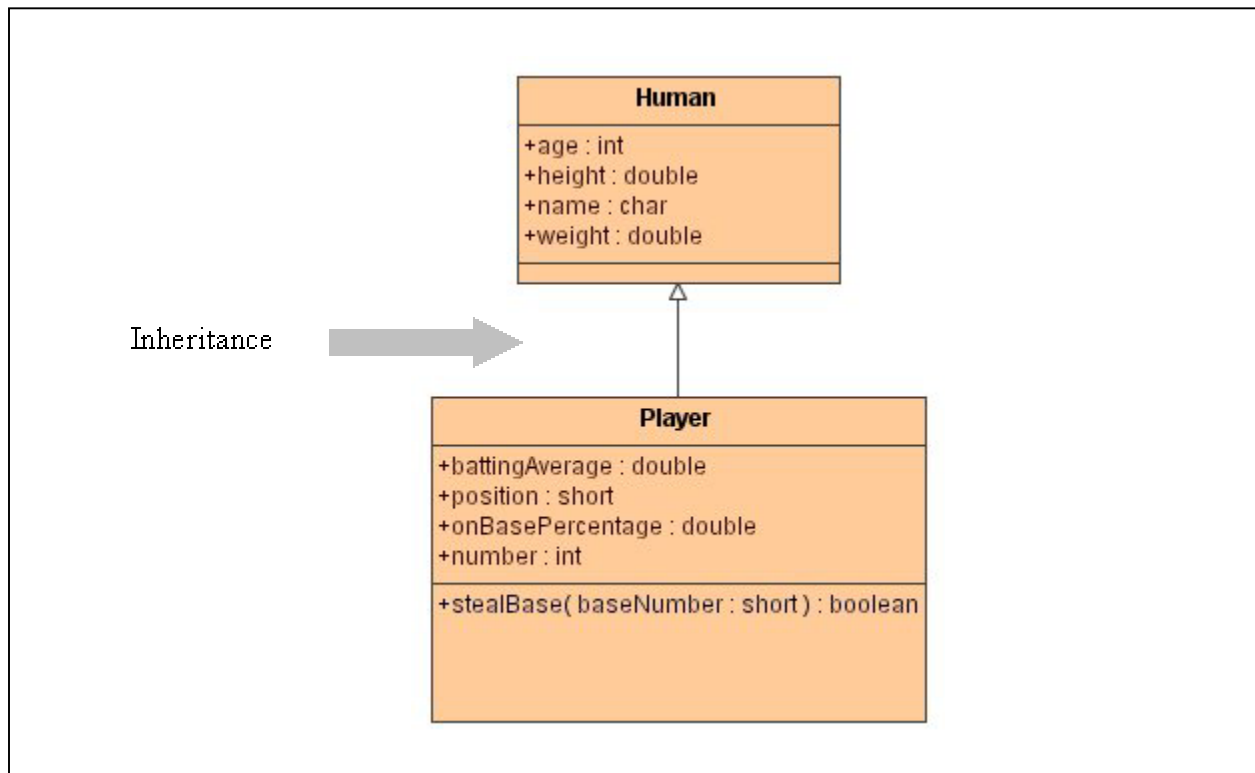


Figure 3-4. Inheritance.

Abstraction is defined as a relationship between two descriptions of the same thing. Figure [3-5](#) graphically depicts this concept as both the Coach and Player are of type Human; therefore, a coach “IS-A” Human. Figure [3-6](#) denotes the composition by having the closed diamond on the class that contains the other class. It follows that Figure [3-6](#) depicts a Player class containing Bat, Glove, and Uniform classes, and a coach HAS-A uniform.

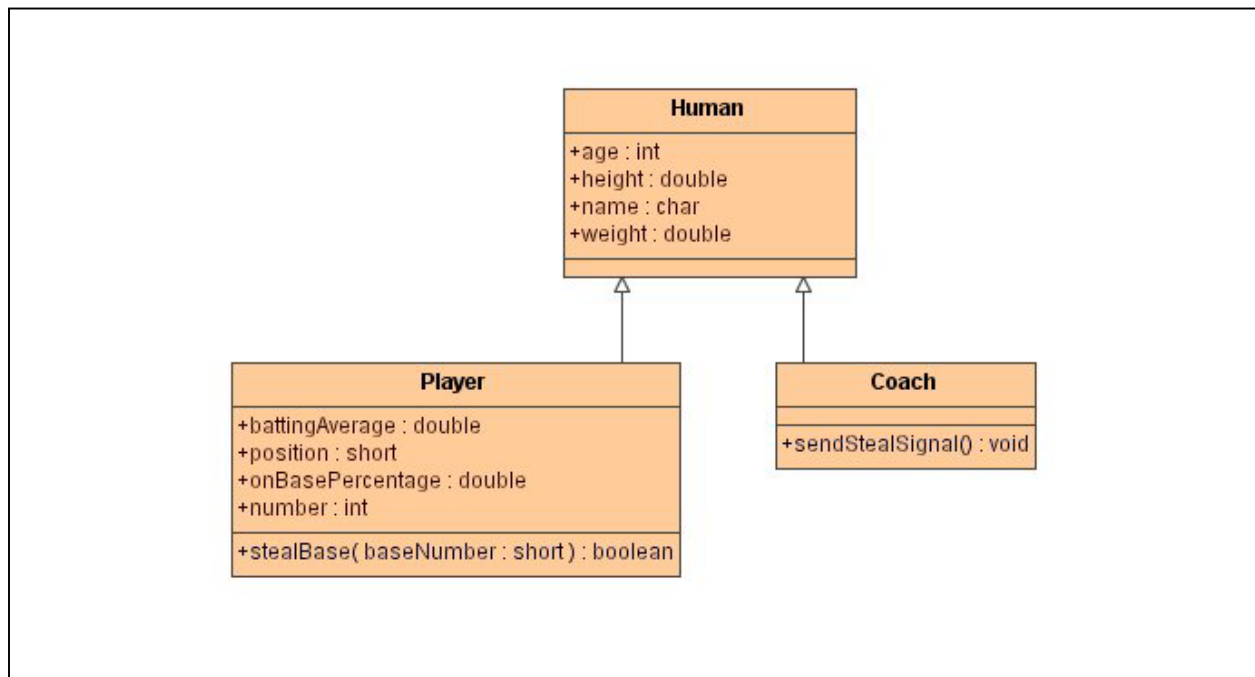


Figure 3-5. Abstraction.

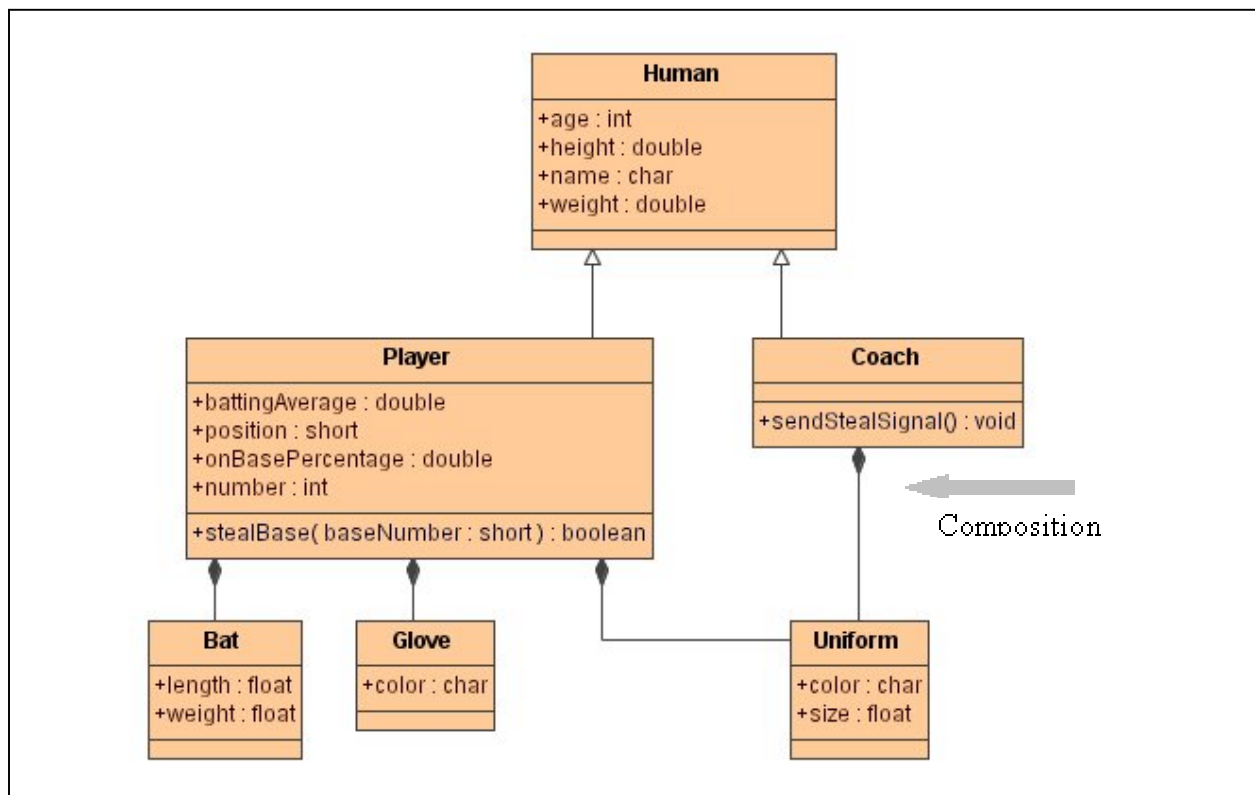


Figure 3-6. Composition.

3.2.4.3 Sequence diagrams. A sequence diagram is an interaction diagram that depicts information flow between software objects, or instance of a class, in a sequenced manner. The information sent between objects is referred to as messages and are depicted by straight lines with arrows to denote direction of message flow. The interaction(s) between objects is shown as a discrete event during the execution of the system. The diagram is configured such that the objects are listed on the top row with vertical lines drawn to the bottom of the diagram under each object. The vertical lines are sometime referred to as lifelines as the execution time progresses from the top to the bottom of the diagram. To summarize, the sequence diagram is a dynamic collaboration diagram that represents the information flow between objects in a sequenced manner. Figure 3-7 is a simple sequence diagram.

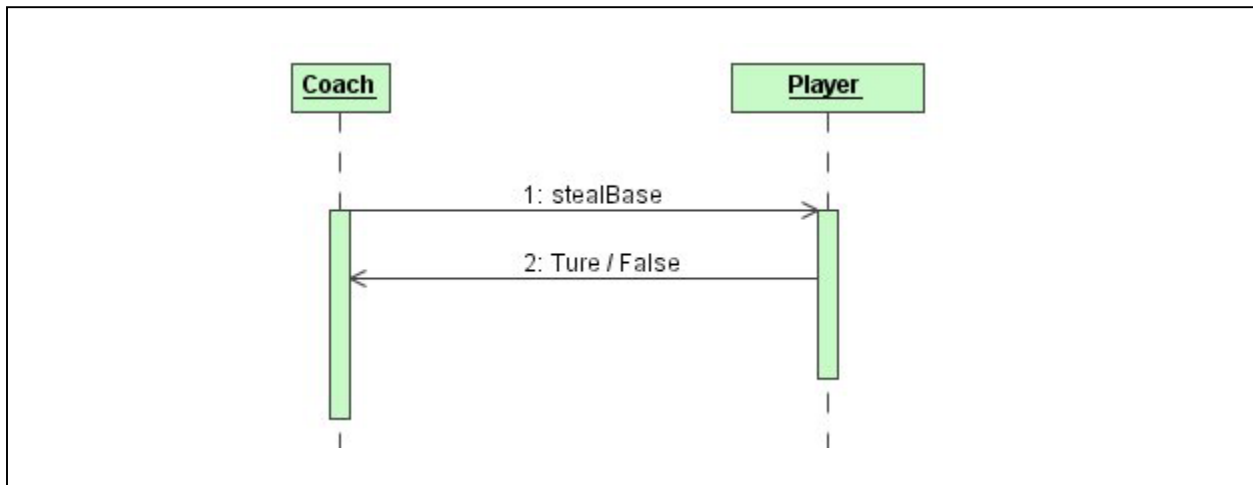


Figure 3-7. Sequence diagram.

3.2.4.4 Deployment diagrams. A deployment diagram represents the configuration of the system's hardware and software components, which are referred to as nodes. This type of diagram contains the connections between the nodes and a description of the connections. The deployment diagram is diagram is useful in presenting the envisioned system configuration and is helpful in understanding potential constraints levied on the system developers. Figure [3-8](#) is a simple deployment diagram.

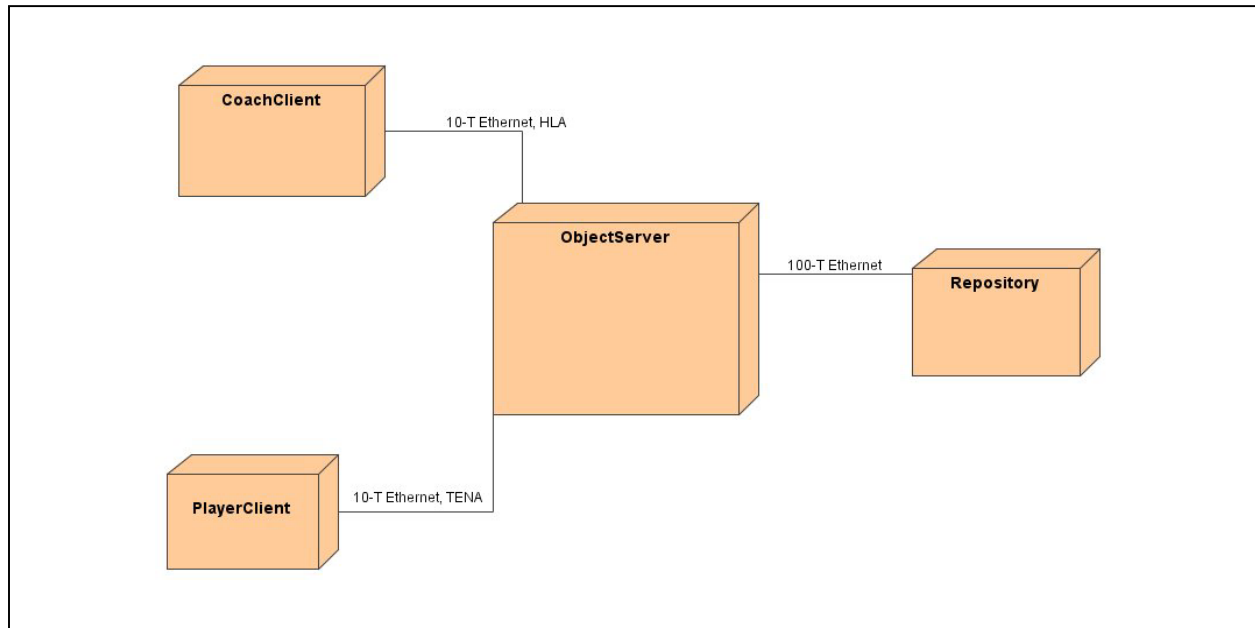


Figure 3-8. Deployment diagram.

3.2.5 Object Model metadata. Metadata is data that is associated with an OM that puts the OM definition into context. Examples of OM metadata could be:

- a. OM Name - XYZradar
- b. Point of contact information - Mr. John Doe with phone, email, organization, and other information
- c. Use Case Name - Range XYZ radar utilization
- d. Range resource represented - XYZ radar system
- e. Fidelity of the application - low, medium, or high
- f. Purpose of OM - to develop an interface to XYZ radar for other ranges to utilize XYZ radar

Figure [3-9](#) depicts the Developmental Test Command (DTC) Integrated Level Hierarchy (ILH) that has been standardized as the metadata for the Virtual Proving Ground (VPG) program. The ILH was developed at the Redstone Technical Test Center (RTTC) and is currently the standard data representation schema used at DTC ranges. While this extensive schema is not necessary for our metadata definition, it is presented here to reinforce the importance of maintaining a consistent set of information that describes the asset in question.

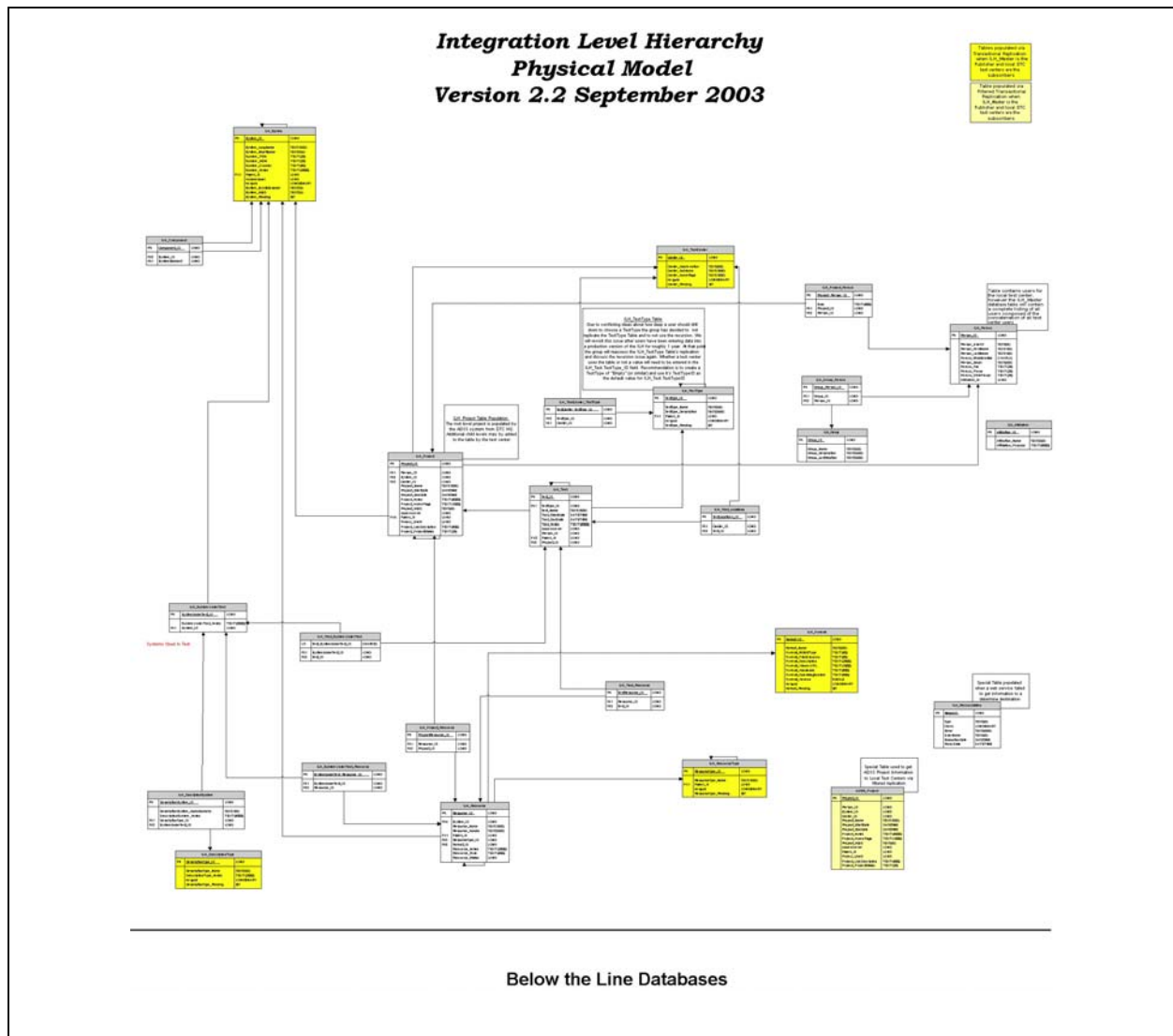


Figure 3-9. The Developmental Test Command (DTC) integration level hierarchy.

3.2.6 Object Model metamodel. A metamodel is a “model that defines a model” and can be considered OM metadata. While this definition may seem to be trivial and almost useless, it is in fact very significant to OM discussions. Another way of defining metamodel is by stating that a metamodel describes the capability of another model. The following are examples of metamodels in use.

- a. C++ metamodel - Classes, structs, multiple inheritance, composition, generics, functions, methods, operators, fundamental data types, exceptions, etc.
- b. Java metamodel - Classes, Interfaces, and exceptions. Doesn't support structs, functions, generics or multiple inheritance
- c. Common Object Request Broker Architecture (CORBA) metamodel - Interfaces, structs, valuetypes, sequences, enumerations, and so forth (doesn't support classes).

- d. High Level Architecture (HLA) metamodel - Classes (as objects), interactions, attributes, single inheritance. Doesn't support interfaces, composition, functions, methods, etc.

One example of a metamodel is the Test and Training Enabling Architecture (TENA) metamodel. The TENA metamodel (see Figure 3-10 directly below) will support classes, local classes, inheritance, composition, interfaces, messages, and other architectural elements. Another metamodel, the HLA, is depicted in Figure [3-11](#).

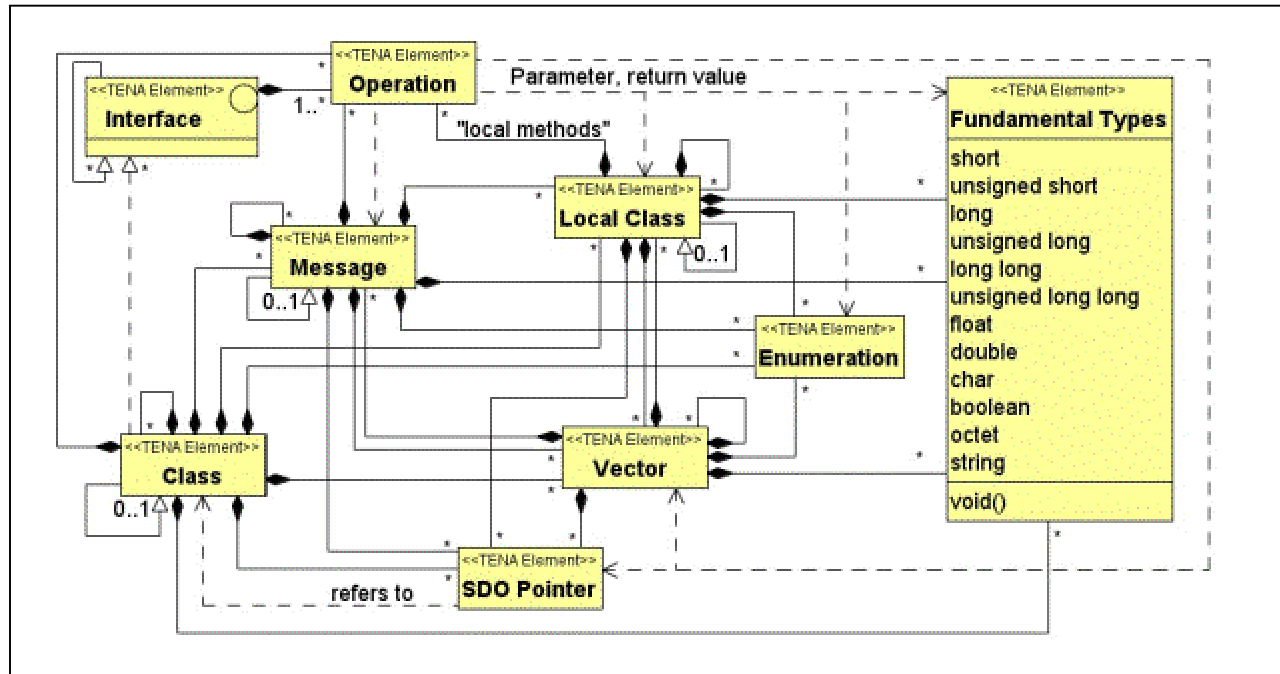


Figure 3-10. TENA metamodel.

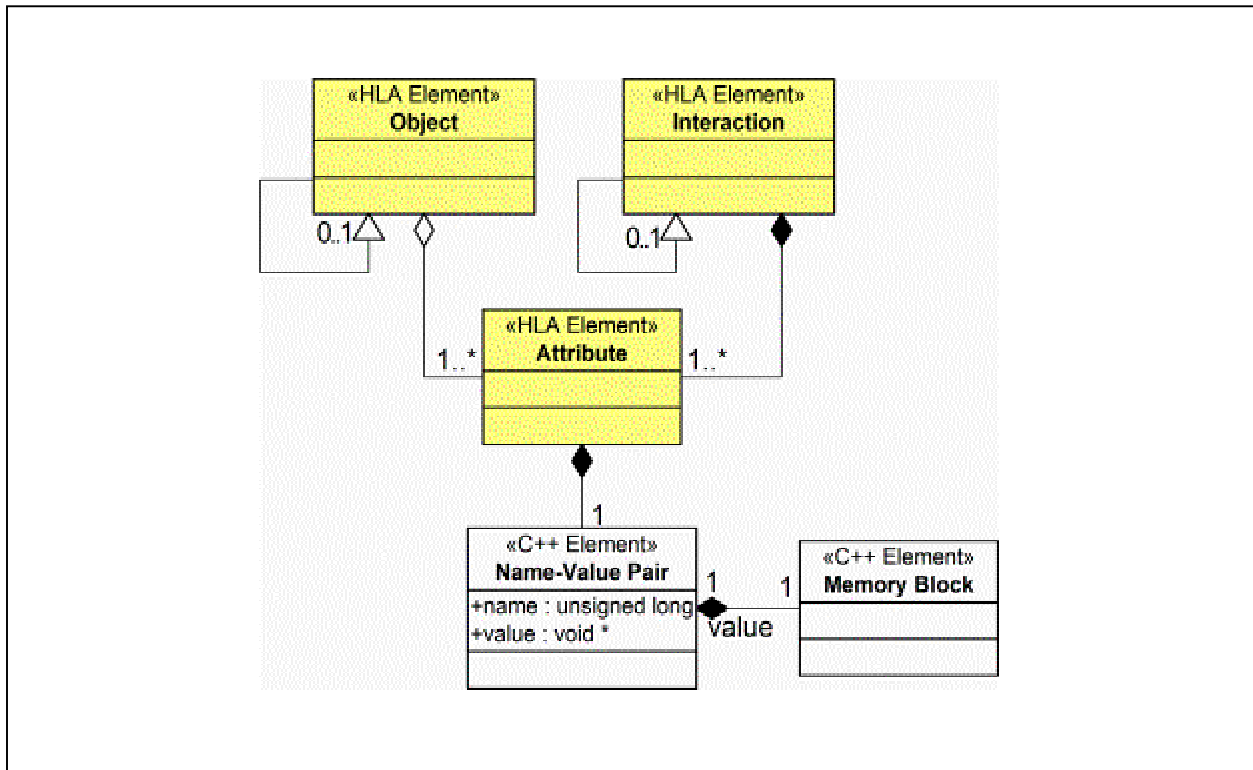


Figure 3-11. HLA metamodel based on the Object Model template.

The reason the metamodel definition is critical is that the metamodel lists all the features of the over-arching system or architecture that the software was designed to use. For example, if an OM had been developed using the HLA metamodel, the OM would never contain the element SDO Pointer because the HLA doesn't utilize the concept of SDO Pointer. It is very important for the RCC to understand the impact of metamodels on OM definitions. We need to ensure that the OM being developed is compatible with the specific architecture metamodel(s) in which the OM will be used. The Range Commanders Council must appreciate the potential conflicts with other OMs designed with a different metamodel and therefore should adopt acceptable metamodel(s) and require OM submissions to follow them.

3.3 Deliverables For Task 2

3.3.1 Proposed OM submission package format. When a user submits an OM to the RCC DR&CG as a potential candidate for standardization, the submission needs to be presented in a package. A package contains several items and is required to give the DR&CG a complete set of information regarding the OM submission. Before an OM is accepted into the OM standardization process, the entire package must be completed. The OM submission package must contain:

- The OM metadata - The metadata fields and structure must be provided to put the OM definition and submission into context.
- The OM - The OM must be graphically depicted using UML notation as a standard class diagram.

- c. Use Case - While in UML a Use Case is only one of many diagrams, a Use Case is referred to herein as several UML diagrams. A Use Case is either a UML-based Use Case model or the FI2010 Use Case template. Note: If UML diagrams are used, the Use Case, sequence, and deployment diagrams are required.
- d. Metamodel - A graphical or textual representation of the metamodel used during the OM definition.

Unless otherwise specified, the OM and Use Case diagrams shall be presented in the UML Standard XML¹ Metadata Interchange (XMI) 1.0 Format. This format allows for diagram interchange between various UML tool programs. As standards evolve, it is recommended that the DR&CG adjust this requirement to meet new standard definitions. In addition, the JPEG format is required for quick-look capability

The OM submission package may either be presented via e-mail to an address designated by the DR&CG or via a DR&CG-supported on-line submission process.

Spring City Solutions, Inc. has implemented a prototype of the OM review process as defined by DR-31 Task 1 that includes the OM package submission process. This implementation demonstrates the feasibility of the OM standardization process as defined by the DR-31 Task 1 and 2 deliverables. An example of this prototype on-line OM submission and management process is presented in [paragraph 3-4](#) below.

3.3.1.1 Metadata. The metadata shall be filled in via the DR&CG-sponsored on-line submission process or filled in and submitted via Microsoft Word Format. Metadata fields are defined in [Appendix A](#).

3.3.1.2 Object Model. The submitter shall provide the OM in JPEG and XMI format. While the JPEG format supports a quick-look analysis, the XMI format allows the DR&CG personnel to import the diagrams into their UML tool for more extensive investigation.

3.3.1.3 Use Cases. The OMG clearly and consistently defines the process, tools, components, and diagrams to generate a system using UML. Therefore, a UML-based Use Case diagram, and other supporting UML diagrams, should be acceptable *as input for an OM submissions in the XMI format*. The minimum required UML diagrams for each use OM submission include:

- a. Use Case diagrams.
- b. Sequence diagrams.
- c. Deployment diagrams.

Additional diagrams that may be included in the Use Case, but are not required, are:

- d. Collaboration diagrams
- e. State diagrams
- f. Implementation diagrams

¹ Extensible Markup Language

However, many in the RCC community are either unaware or unable to spend the time to fully embrace and implement the Use Case definition process. Therefore the pure UML definition of the Use Case development process should not be the lone Use Case submission format for OM submission to the RCC DR&CG standardization. A range-specific Use Case format generated by the Foundation Initiative 2010 project should also be endorsed as a submission format as it better reflects the range community terminology, process, and communication constructs.

Additional guidance is contained in the appendices as follows:

[Appendix B](#) - FI2010 Use Case Templates

[Appendix C](#) - FI2010 Use Case Instructions

[Appendix D](#) - Example Use Case: Radar (hypothetical scenario)

3.3.1.4 Meta-model. As defined earlier, the metamodel is the model that describes the model of the architecture, like TENA, or the single language the system was written in, like C++. Therefore the user must, at a minimum, state the metamodel being used, but preferable provide a graphical representation of the metamodel in JPEG format.

3.4 **Prototype On-Line OM Submission Process**

As an unfunded feature, Spring City Solutions, Inc. developed an on-line OM submission and management tool suite to provide the feasibility of the process and submission formats defined in DR-31. While this tool is only a prototype, it demonstrates the concept of managing OM submissions, further development of this working prototype is possible if the DR&CG wishes to pursue this activity. A sample of the prototype system user screens can be seen at Figures 3-12 through Figure 3-29.

- Figure [3-12](#). Website home page
- Figure [3-13](#). Website login
- Figure [3-14](#). OM submission
- Figure [3-15](#). OM submission - completed
- Figure [3-16](#). DR&CG Chair login
- Figure [3-17](#). Review Team created by Chair
- Figure [3-18](#). Review Team created by Chair - completed
- Figure [3-19](#). Review Team login
- Figure [3-20](#). Review Team comment
- Figure [3-21](#). Review Team comment - completed
- Figure [3-22](#). Working Group creation by Chair
- Figure [3-23](#). Working Group creation by Chair – completed
- Figure [3-24](#). Working Group login
- Figure [3-25](#). Working Group choices
- Figure [3-26](#). Working Group comments
- Figure [3-27](#). Working Group acceptance of topic
- Figure [3-28](#). Review Team monitoring working group status
- Figure [3-29](#). DR&CG Chair acceptance



Figure 3-12. Website home page.

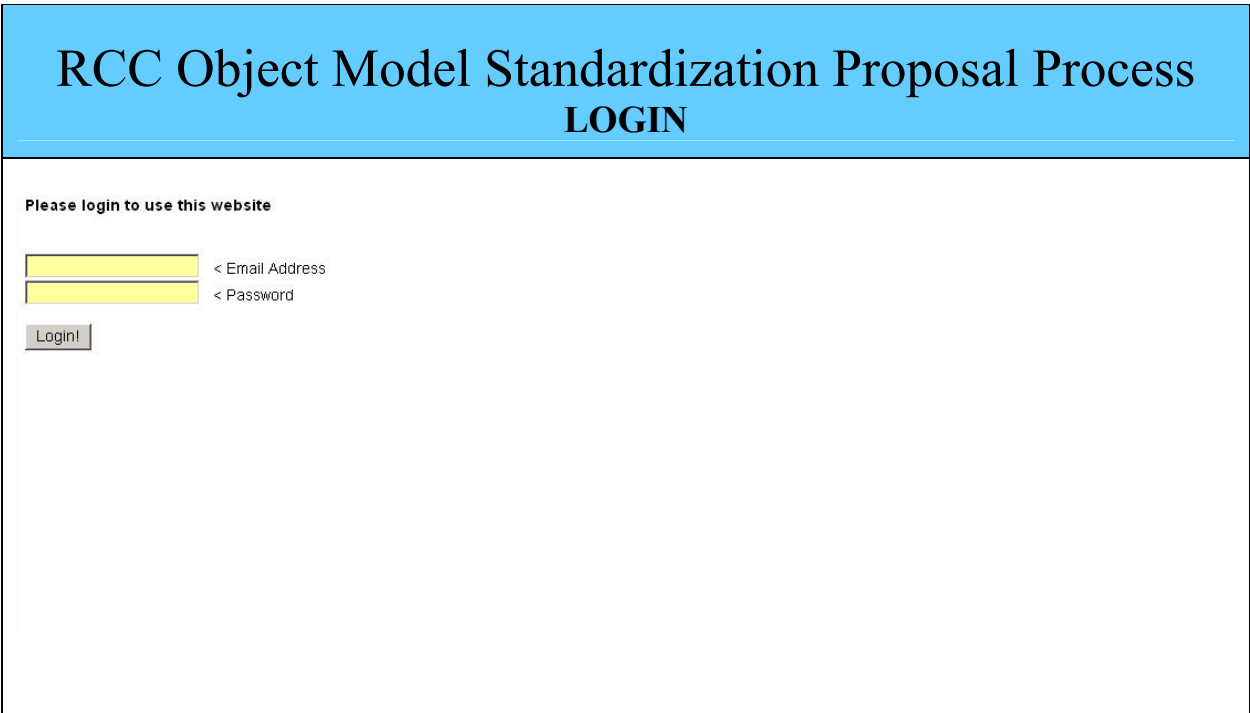


Figure 3-13. Website login.

RCC Object Model Standardization Proposal Process

SUBMIT OM

Step 1: Object Model Package Submission

- 1) Object Model Name:
- 2) Please attach a zip file containing the files associated with this Object Model

3) Insert Metadata for this OM Submission Package

Name:

Organization:

Mail Stop:

Phone Number

Fax Number:

Email Address:

Object Model File Name(s):

Range Object(s) Defined:

Meta-model OM Designed to Support

Object Names Submitted for Standardization:

Purpose of OM:

Figure 3-14. Object Model submission.

RCC Object Model Standardization Proposal Process

SUBMIT OM

Object Model Submitted to the RCC Chair for processing!

>> Object Model Name: Radar

>> Package File Renamed to: Radar-TENA_Radar.zip

Figure 3-15. Object Model submission - completed.

RCC Object Model Standardization Proposal Process

DR&CG CHAIR

DR&CG Chair Creates Review Team

Welcome DR&CG Chair!

As the DR&CG Chair, you may:

→ Create a Review Team for an Object Model below:

→

→ Accept or Deny an Object Model Submission below:

→ →

→ [View a status list of all submitted Object Models](#)

OM NAME	OM Status
GPS_OM	→ REVIEW TEAM
Radar	→ PROPOSED
Widget	→ REVIEW TEAM

Figure 3-16. DR&CG Chair login.

RCC Object Model Standardization Proposal Process

DR&CG CHAIR

DR&CG Chair Creates Review Team

Welcome DR&CG Chair!

→ [View a status list of all submitted Object Models](#)

You may now subscribe members to the Review Team for **Radar**

Chair, DR&CG ▾

→

Add To Review Team

Currently subscribed review team members:

- Cox, Seth <[Unsubscribe](#)>
- Lessmann, Kurt <[Unsubscribe](#)>

• When you have finished subscribing users, you can create the review team by clicking the button below...

→

Create Review Team

Figure 3-17. Review Team created by Chair.

RCC Object Model Standardization Proposal Process

DR&CG CHAIR

DR&CG Chair Creates Review Team

Welcome DR&CG Chair!

→ [View a status list of all submitted Object Models](#)

Review team has been created for this object model!

Follow these login instructions to access the Review Team area:

- Go to [reviewTeam.php](#)
- Enter your email address and choose the Object Model you wish to review (note: you must have a subscribed email address to access any particular Object Model)

Figure 3-18. Review Team created by Chair - Completed.

RCC Object Model Standardization Proposal Process

REVIEW TEAM

Review Team Accept/Deny or Request Revisions

Welcome to the Review Team access page

If you are subscribed as a Review Team member for an Object Model, you may access that OM by entering the subscribed email address along with choosing the OM you wish to review

Subscribed email address:

Object Model:
 →

Figure 3-19. Review Team login.

RCC Object Model Standardization Proposal Process

REVIEW TEAM

Review Team May Accept/Deny Object Model or Request Revisions

Welcome to the Review Team access page.

You are currently working with the Proposed Object Model **Radar**

Please review the documents contained in the Object Model package. You may then decide to Accept or Deny this proposal or create a Standard Proposal Working Group to consider a revision.

(right-click to download Object Model package)
→ [Radar-TENA_Radar.zip](#)

You may:

- » [Accept this proposal and send back to DR&CG Chair for processing](#)
- » [Deny this proposal and send back to DR&CG Chair for removal](#)
- » [Create A Standard Proposal Working Group For Revision](#)

No working groups have been defined yet.

Add comment to this Object Model Proposal:

Figure 3-20. Review Team comment.

RCC Object Model Standardization Proposal Process

REVIEW TEAM

Review Team May Accept/Deny Object Model or Request Revisions

Welcome to the Review Team access page.

You are currently working with the Proposed Object Model **Radar**

Please review the documents contained in the Object Model package. You may then decide to Accept or Deny this proposal or create a Standard Proposal Working Group to consider a revision.

(right-click to download Object Model package)

→ [Radar-TENA_Radar.zip](#)

You may:

- » [Accept this proposal and send back to DR&CG Chair for processing](#)
- » [Deny this proposal and send back to DR&CG Chair for removal](#)
- » [Create A Standard Proposal Working Group For Revision](#)

No working groups have been defined yet.

Add comment to this Object Model Proposal:

Submit Comment

Comments submitted for this Object Model:

Comment submitted by Kurt Lessmann on April 16th, 2004, 3:14pm
I think this is a wonderful design!

Figure 3-21. Review Team comment – completed.

RCC Object Model Standardization Proposal Process

You are currently working with the Proposed Object Model **Radar**

For updates and changes to this Object Model submission, you may create working groups to review the changes.

Please enter a short description of the function for this Working Group:

Review Antenna Class

Please enter the details for this Working Group submission:

Should aperture be contained in this class?

→ Step 2: Assign Users to This Working Group

Figure 3-22. Working group creation by Chair.

RCC Object Model Standardization Proposal Process

A working group has been created for this object model!

Follow these login instructions to access the Working Group area:

→ Go to [workingGroup.php](#)

Figure 3-23. Working Group creation by Chair - completed

RCC Object Model Standardization Proposal Process

WORKING GROUP

Welcome to the Object Model Working Group access page

If you are subscribed as a Working Group member for an Object Model, you may access that group by entering your subscribed email address below. You will then be shown your subscriptions to choose from.

Subscribed email address:

→

Access Working Group Subscriptions

Figure 3-24. Working Group login.

RCC Object Model Standardization Proposal Process

WORKING GROUP

Welcome to the Object Model Working Group access page

You are subscribed to these Object Model Working Groups. You may select any of these to enter the Working Group area for that Object Model.

Description	Associated Object Model
Review Widget attribute "stuff"	Widget
Review Antenna Class	Radar

Figure 3-25. Working Group choices.

RCC Object Model Standardization Proposal Process

WORKING GROUP

Working Group May Make Comments, Accept, Deny or Modify The Revision Request

Welcome to the Review Team access page.

You are currently working with the Proposed Object Model **Radar**

(right-click to download Object Model package)

→ [Radar-TENA_Radar.zip](#)

You may:

- » [Accept this proposal and send back to Review Team for processing](#)
- » [Deny this proposal and send back to Review Team for removal](#)

Below is the described function of this Working Group:

Should aperture be contained in this class?

Add comment to this Working Group:

Submit Comment

Comments submitted for this Working Group:

Comment submitted by Seth Cox on April 16th, 2004, 3:24pm
Based on my experience, aperture is needed.

Comment submitted by Kurt Lessmann on April 16th, 2004, 3:24pm
Yes, I agree - let's go with it.

Figure 3-26. Working Group comments.

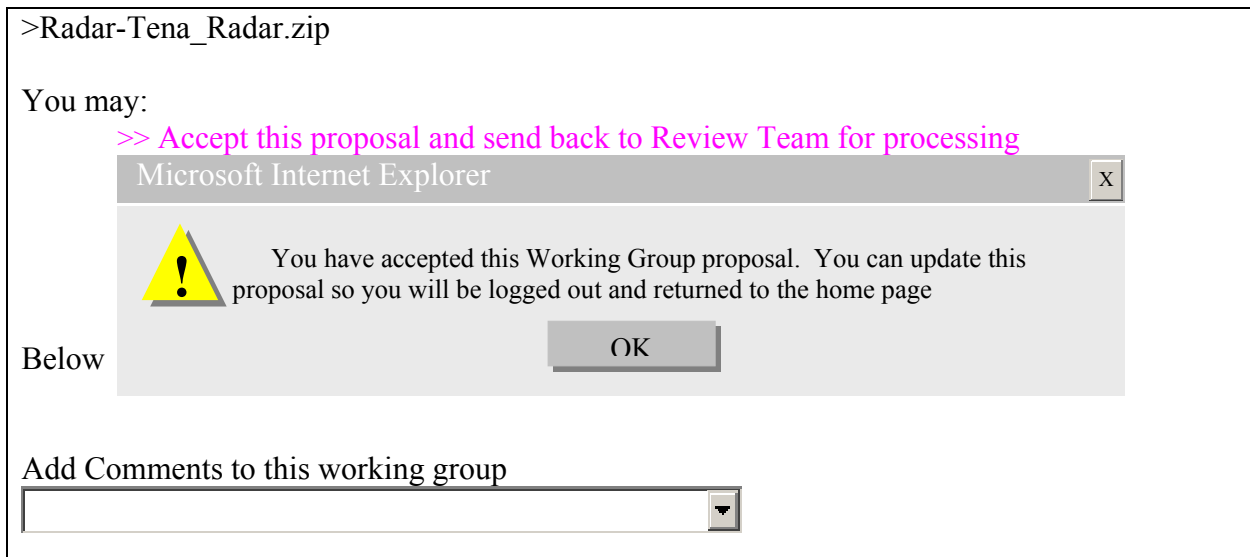


Figure 3-27. Working Group acceptance of topic.

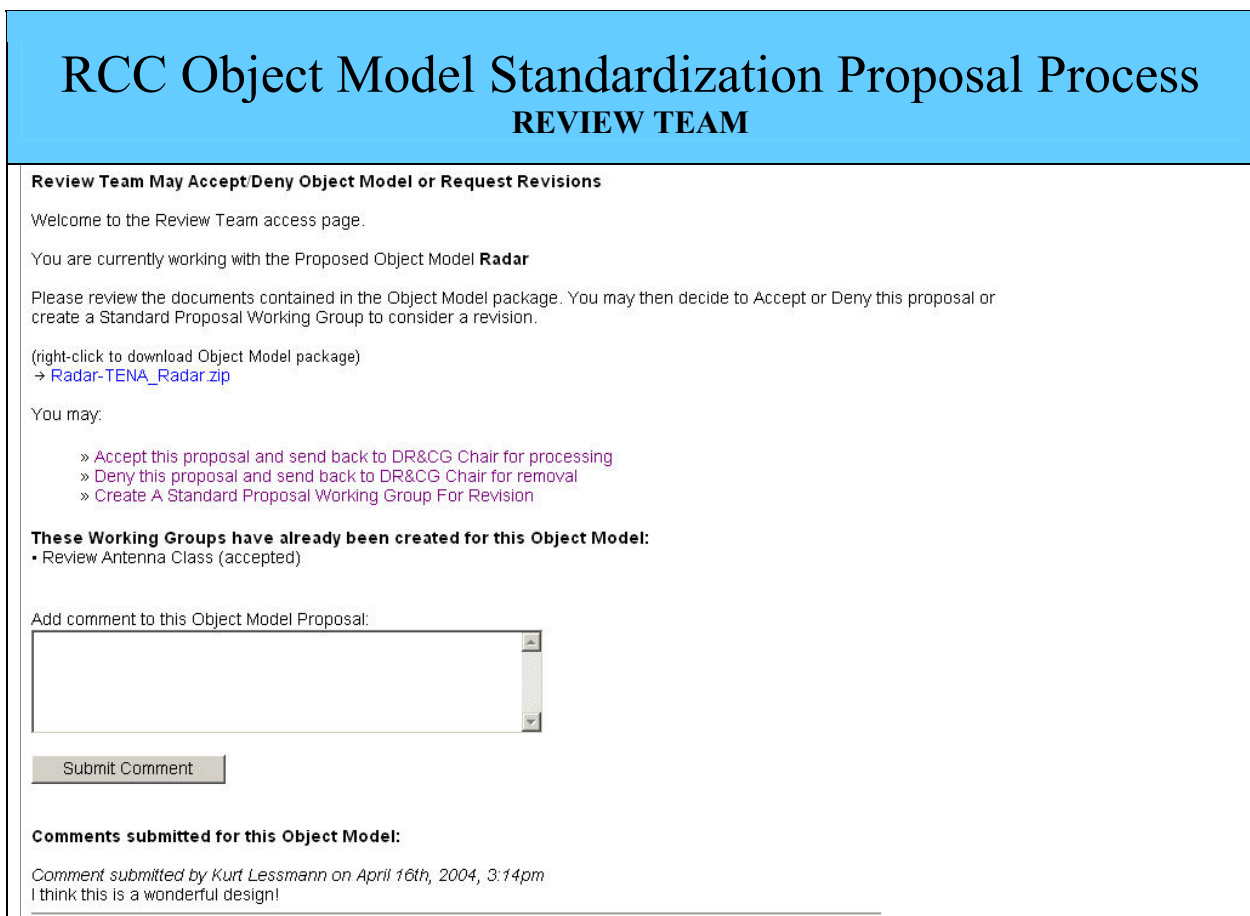


Figure 3-28. Review Team monitoring working group status.

RCC Object Model Standardization Proposal Process

REVIEW TEAM

DR&CG Chair Creates Review Team

Welcome DR&CG Chair!

As the DR&CG Chair, you may:

→ Create a Review Team for an Object Model below:

→ *There are currently no Object Model submissions without a review team assigned.*

→ Accept or Deny an Object Model Submission below:

→ →

→ [View a status list of all submitted Object Models](#)

Figure 3-29. DR&CG Chair acceptance.

APPENDIX A

METADATA FIELDS

The recommended metadata fields are shown below (Required = *)

Point of Contact (POC) POC Information

- *Name
- *Organization
- *Mail stop
- *Phone number
- *Fax number
- *Email address:

Object Model (OM) Submission Information

- OM Submission Package Information
 - * Submitted Zip file Name
 - Version number
 - *Object name(s) submitted for standardization
 - *Purpose of OM - Text
 - *Range Object(s) defined
 - *metamodel utilized during development
 - Re-use of Existing OMs
 - Existing OM Name
 - Submitted OM Name
 - Developing Organization
 - OM modifications
- OM Submission Package
 - Use case file name(s)
 - In FI2010 template
 - UML Diagrams
 - Object Model file name(s)
 - Graphics file (JPEG, GIF,etc.)
 - XMI file – including XMI version number

OM Definition Testing

- Range event(s) OM supported – Fill out 1 for each event
 - *Customer
 - *Date
 - *Event Description - Text
 - *Participating Locations - Text
 - Network and computer configurations - Text
 - Number of:
 - *Range instrumentation/assets
 - *live participants – Non instrumentation
 - *virtual test applications

- *Constructive applications
 - Number of instances of:
 - *Range instrumentation/assets
 - *live participants– Non instrumentation
 - *virtual participants
 - *Constructive participants
 - Time duration of event
 - Number of test runs
 - General Event Comments
- General Submission Comments

APPENDIX B

FOUNDATION INITIATIVE 2010 USE CASE TEMPLATES

The following Use Case “Worksheets” are provided below as:

Figure [B-1](#) - Test and Training Enabling Architecture (TENA) Use Case basic information

Figure [B-2](#) - TENA applications

Figure [B-3](#) - Events

Figure [B-4](#) - Information flow

Figure [B-5](#) - Object Model

Figure [B-6](#) - Other relevant information

1.1 TENA Use Case Basic Information			
1.1.1 TENA User Name *			1.1.6 Background*
1.1.2 Use Case Name *			
1.1.3 Basic/Complex *			
1.1.4 Customer			
1.1.5 Other Use Cases Referred to			
1.1.6 Key Words			
starred (*) blocks are required			

Figure B-1. TENA Use Case basic information.

1.2 TENA Applications													
Application Information *					1.2.6 Operating System (Type & Version) *	1.2.7 Computer Type *			1.2.8 Network Connection (Type & Speed) *	1.2.9 Input Ports (USB, Keyboard, etc.)	1.2.10 Output Ports (Printer, etc.)	1.2.11 Notes	1.2.10 Connectivity Diagram *
1.2.1 ID *	1.2.2 Application Name *	1.2.3 Programming Language *	1.2.4 Compiler *	1.2.5 Threading Info *		Name	CPU *	RAM (MB) *					
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													

Figure B-2. TENA Applications Worksheet.

	A	B	C	D
1	1.3 Basic Course of Events			
2	1.3.1 Action ID *	1.3.2 Action or Event *	1.3.3 IKE 2 Expected Functionality During this Action	1.3.4 Process Diagram
3	1			
4	2			
5	3			
6	4			
7	5			
8	6			
9	7			
10	8			
11	9			
12	10			
13	11			
14	12			
15	13			
16	14			
17	15			
18	16			
19	17			
20	18			
21	19			
22	20			
23	21			
24	22			
25	23			
26	24			
27	25			

Figure B-3. Events Worksheet.

1.4 Information Flows												1.4.13	
1.4.1 ID *	1.4.2 Name *	1.4.3 Type (Include Data Type and Interface Standard, if any) *	1.4.4 Originating Application *	1.4.5 Receiving Application *	1.4.6 Gateway Required *	1.4.7 Size (bytes) *	1.4.8 Frequency (Hz) *	1.4.9 Latency (seconds) *	1.4.10 Real Effect or Reliability *	1.4.11 Is Ordering Critical? *	1.4.12 Required Data Access Method? *	Information Flow Diagram	
												(No required chart, or any other type of chart showing Information Flows)	
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													
31													
32													
33													
34													
35													
36													
37													
38													
39													
40													
41													
42													

Figure B-4. Information Flow Worksheet.

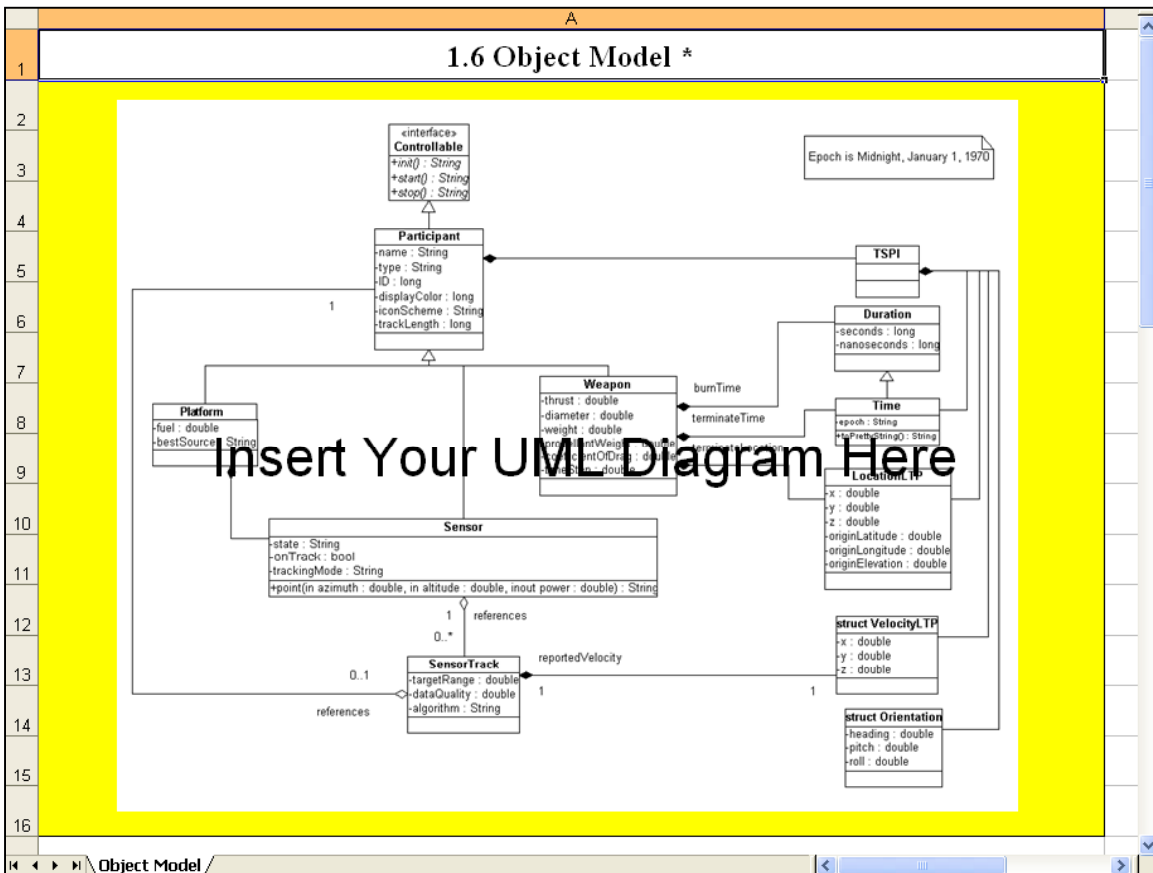


Figure B-5. Object Model Worksheet.

1.7 Other Relevant Information as Required - Optional	
Note #	Note Comment
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Figure B-6. Other Information Worksheet.

APPENDIX C

FI 2010 TENA USE CASE INSTRUCTIONS

1.1 Introduction

TENA Use Cases are intended to help the TENA Middleware Developer understand the Range Community's requirements for creating the TENA Middleware. The TENA Use Case Template is contained in a series of Excel spreadsheets. This Excel package contains both the templates for defining a Use Case and for executing the Use Case Evaluation Process. This section presents the instructions for defining Use Cases such that the maximum amount of pertinent information is conveyed systematically. [Paragraph 1.2](#) below provides guidance on completing the evaluation templates as part of the Use Case Evaluation Process.

1.1.1 Requirements Analysis Process. Because the process for analyzing TENA requirements is very iterative, it is difficult to show cleanly on a diagram. However, Figure C-1 is provided in an attempt to give the reader a good overview of the process.

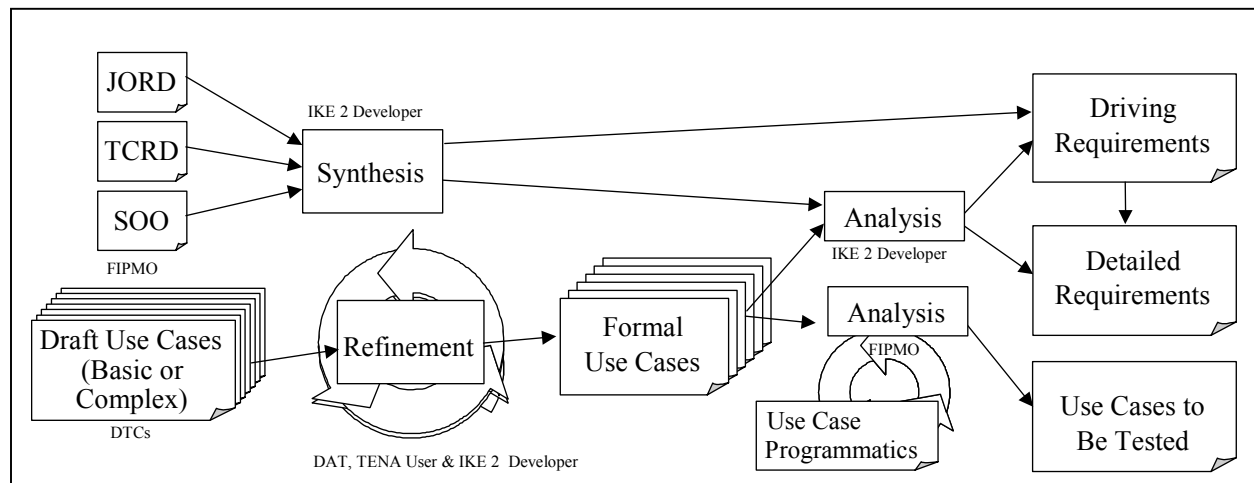


Figure C-1. The Requirements analysis process for the TENA Middleware prototype.

The goal of the current requirements analysis process is to fully understand the requirements for TENA Middleware. These requirements include the “Driving Requirements” which are the high level requirements specifying the essence of the system’s purpose. The requirements also include the “Detailed Requirements” which include a list of all the TENA Middleware functions as well as all the non-functional requirements imposed on the TENA Middleware. Another goal is to produce a series of test cases, based on the Use Cases, in which the TENA Middleware can be tested. Upon receipt of a Draft Use Case, the TENA Development Advisory Team (DAT) and the TENA Middleware Developer, in consultation with the Use Case designers (i.e., the potential TENA Users), will refine each Draft into a formal, more structured Use Case suitable for requirements extraction and the development of the test plan. Both the TENA User and the TENA Middleware Developer will sign off on the final version of the Use Case. This formalization process is intended to make sure the TENA Middleware Developer fully understands what is being presented and that the TENA User agrees with any refinements made to the Use Case by the TENA Middleware Developer.

Use Cases can range from the very “Basic,” in that it contains a very small number of TENA applications performing only a very few functions, to the more “Complex,” containing more than a few applications, operating over an extensive period of time with a large variety of information being interchanged. The TENA Middleware development team requires more basic Use Cases than complex Use Cases. The complex Use Cases should (mostly) be able to be constructed by combining the basic Use Cases.

1.1.2 Purpose of the Use Cases. There are two main purposes for the development of Use Cases. The first purpose of the Use Cases is to educate the TENA Middleware development team on what type of environment exists in the Range Community in which TENA applications using TENA Middleware will be operating. The two key questions that need to be addressed in a Use Case are:

- a. What types of applications exist on the range?
- b. What are the information flows between these applications?

The second purpose of developing Use Cases is to provide the Foundation Initiative 2010 Project Management Office (FIPMO) with information that can be used to test the TENA Middleware software when it has been developed.

1.1.3 Tips for good stories. In general, there should be a large number of small basic Use Cases. These can then be combined later into more extensive complex Use Cases that can be used for testing. The basic Use Cases should:

- a. Be small - involve only two to three systems, with only a few types of information being exchanged.
- b. Be instructive - assume the TENA Middleware development team is not knowledgeable about range activities.
- c. Include processes or problems that are both easy to do as well as difficult to do. Not every Use Case must push the performance envelope or demonstrate an extremely difficult function - some should tell about very basic range operations so the development team can get a feel for those processes as well.
- d. Include both processes and applications that currently exist on the range as well as processes and applications that the users hope will be made possible with the introduction of TENA in the next two years. There are thus two kinds of stories told in these Use Cases (“historical fiction” and “science fiction”) depending on whether you want to show functionality that already exists on the range that TENA Middleware must accommodate, or whether you want to illustrate functionality not currently possible on the range, but that TENA Middleware should address.
- e. Cover a wide variety of scenarios, the more the better.

1.2 Use Case Template Instructions

The items shown in the following sub-paragraphs are important for each Use Case. The actual format for the text of each Use Case is hyperlinked to an associated Excel spreadsheet at [Appendix D](#). Diagrams showing connectivity, process, and information flow (described below) are very important and should be included in the Use Case. UML diagrams ([paragraph 3.2.4](#)

above) are best, so it is suggested that they be used, but not everyone should have to be a UML expert to produce a good and useful Use Case. Therefore, if unfamiliar with UML, the reader is free to use any diagramming technique with which he/she is familiar. The refinement process will include converting non-UML diagrams into standard UML diagrams. Some information is required and some information is optional. Required information is marked with an asterisk and is shaded in the template. Nevertheless, if including a required item makes no sense, then the item should be omitted. Similarly, if there is crucial information you need to convey that has no natural place in the template, include it on the last page of the template entitled “Other relevant information as required.”

1.2.1 Basic information. This section gives the name of the Use Case, a summary, and other important administrative shown below. A sample spreadsheet is at Figure [D-1](#).

- a. TENA User Name - Name of the organization submitting the TENA Use Case.
- b. Use Case Name - The name should be unique and descriptive.
- c. Basic/Complex - State here whether this Use Case is a basic Use Case or a complex Use Case.
- d. Customer - By default this may be your range/organization, but it may be some other range or another investment project, or it might be the range user.
- e. Other Use Cases referred to - In this Section, if this Use Case is a complex Use Case, please list all the other Use Cases out of which this Use Case is constructed.
- f. Background - Describe in this section any background information about your range, the included systems, or the processes and functions that will be implemented in the Use Case that you think the reviewing team might need to understand the Use Case.
- g. Use Case Overview - A short overview should be written in a few sentences describing the Use Case and its primary purpose.
- h. What this Use Case adds and why you are including it - These questions should be concisely answered in this section.

1.2.2 TENA applications. This section should contain a list of the TENA applications (running the TENA Middleware) present in the Use Case. Assume for this section that there are no extant TENA-created tools, so include all tools you may need for these Use Cases as applications in the list. Sometimes multiple applications run on a single computer. Please list each of these applications separately, and include the computer information for each of them. Use the “Notes” section to state that they share a computer. The specific TENA information requirements are shown below. A sample spreadsheet is at Figure [D-2](#).

- a. ID - A consecutive counter for each application (e.g., 1,2,3...)
- b. Application Name - The common name of the application used on your range (e.g., “SimDis”)
- c. Programming Language - List the computer programming language or languages used by the particular application. The most important programming language is that language used for the reading and writing of information to the network. List this language first. Then list any other languages that may be used (e.g., C++).

- d. Compiler - List the compiler (or compilers) used to create this application (e.g., MVC++ 7.0). For interpreted languages (such as Visual Basic or Java), indicate the version number of the interpreter (e.g., VB5.0, Sun JVM 2.0).
- e. Threading Information - Indicate whether this application is single threaded, multithreaded, or multi-process.
- f. Operating System (OS) - Indicate which operating system (including version) this application runs on. List multiple OSs if the application is portable between multiple OSs. For interpreted applications that run on virtual machines that are portable between platforms (e.g., Java applications), list the operating system that the code actually runs on in practice on your range. An example would be Sun Solaris 8.0 for Sparc.
- g. Computer Type - What kind of computer does the application run on? List the name of the computer (e.g., UltraSparc 180), the CPU type and speed (e.g. Dual Intel Pentium III Xeon 1 GHz), and the amount of memory that the system has (e.g., 1GB).
- h. Network Connection - Indicate the type of and speed of the network connection (or connections) that this computer uses to communicate with other computers in the Use Case (e.g., 100Base-T, 10Mbps shared, or Switched Gigabit Ethernet).
- i. Notes - Make any remarks in this area about the applications that are important to your Use Case, but that don't fit in anywhere else. An example of something to include here is that two applications run on the same computer.
- j. Connectivity Diagram - There should be a diagram showing all the TENA applications and the connectivity between each. This diagram can be pasted directly into the template or included as a separate PowerPoint or Visio diagram. A simple non-UML example might be as shown in Figure C-2:

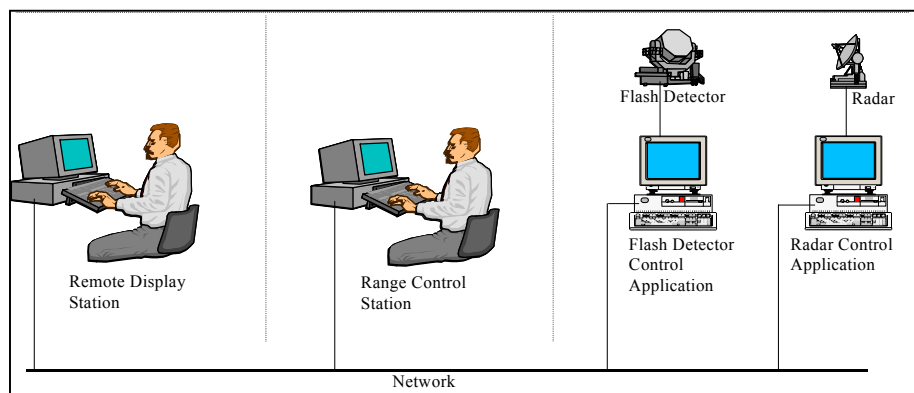


Figure C-2. Connectivity diagram example.

1.2.3 Basic course of events. This section should contain a temporal description or narrative of the events that occur during the Use Case. The information requirements are shown below. A sample spreadsheet is at Figure [D-3](#).

- a. Action ID - A consecutive counter for each action (e.g., 1,2,3...).
- b. Actions or Events - This is the meat of the Use Case and should read like a story telling the reviewers what happens when and how and to whom. Each action in the

Use Case should be presented in its own cell. The cells expand so do not be concerned with fitting the description into one or two lines.

- c. TENA Middleware expected functionality for this action - This is an optional column. For each action, list what you believe to be the appropriate action taken by the TENA Middleware. How is TENA Middleware helping you make the Use Case work during this action?
- d. Process Diagram (Optional) - If it helps, there should be a process diagram, showing system activities, causal relationships, and the progression of time.

1.2.4 Information flows. At its core, TENA Middleware is a software infrastructure that facilitates information exchange between TENA applications. Thus, describing the type of information flows and the direction(s) of the flows is critical in order to extract detailed requirements from the Use Case. In this section of the template, you should list, in a quantitative fashion, every type of information exchange or interaction that takes place in the Use Case. A sample spreadsheet is at Figure [D-4](#).

- a. ID - A consecutive counter for each information flow (e.g., 1,2,3...).
- b. Name - A descriptive name for this information flow (e.g., “radar track data”).
- c. Type - Provide a description of the type of data. There are all types of data (e.g., streamed audio, video, telemetry, command information, position information). Please be as descriptive as possible herein order to understand what is in the data that is being sent. Use as much space as is required to describe the information.
- d. Originating Application: Which application produces this information (cross-referenced back to the previous table)? If multiple applications produce this type of data, there should be **multiple** entries in the table (i.e., the table lists information *flows*, not information *types*).
- e. Recipient Applications - Which application or applications receive this information? List all applications that receive the information and make sure they correspond accurately to the applications listed in the TENA applications spreadsheet located in Figure [D-2](#).
- f. Gateway Required - List the Gateway type required for execution of this information flow. For example, if an HLA application is providing data for this information flow, list “HLA Gateway.” If you are using a TENA to TENA application write TENA.
- g. Size - State how large (in bytes) is one unit of the information (e.g., a 200-byte PDU).
- h. Frequency - List how often this unit of information is produced (e.g., “10 Hz”, or “intermittently,” or “once”).
- i. Latency Tolerance - What are the real get-the-job-done requirements on the amount of end-to-end latency in the delivery of this information? Please be as generous as possible, as latency is the most critical (and thus hard to achieve) performance requirement for the TENA MIDDLEWARE system. Make sure the needs listed here are real mission-critical needs, not just desires (e.g., 100 ms). Also, are there any latency variance (“jitter”) requirements on the information delivery? In other words, you might be able to accept a large latency, say 1 second, but you may require that each time this information is delivered, the latency is relatively close to the same value. So in describing latency requirements, it is important to express the latency as two numbers: a value and a variance, such as 100ms \pm 10 ms.

- j. Best Effort or Reliable - What type of transport mechanism is used (and is required) for this information? Usually information is sent “Best Effort” where there are no rock-solid guarantees that any particular unit of information gets to its intended recipients. Sometimes, however, information needs to be sent in a reliable fashion. In other words, this amounts to asking whether the information should be sent by UDP or TCP. List “Best Effort” or “Reliable” in this box, with the understanding that reliable transport may have significant performance implications.
- k. Is Ordering Critical - In this box tell us whether the recipients can tolerate this information being delivered out of time order. In large distributed systems, sometimes messages can be delivered in a different order than they are sent. Most software implementations can ensure that this rarely happens for quite little cost. However, to guarantee that information is **never** delivered out of order is a very difficult task with significant performance implications, but it can be done if absolutely necessary (and the user wants to pay the performance price for it). Indicate here whether ordering is crucial for this type of information (e.g., Yes or No).
- l. Required During Action Number - List the Action ID from Figure [D-3](#) that corresponds to the sending of this information.
- m. Information Flow Diagram - You should include an (optional) information flow diagram such as an N² chart to illustrate the information flow. Each participant should be included and all types of information between each participant should be listed.

1.2.5 Key functionality required of TENA Middleware that this Use Case demonstrates. Each Use Case has an explicit purpose. It is in this section that you state that purpose explicitly so that there is no doubt in our minds what functionality or performance parameters this Use Case is intended to convey. In addition, provide information on whether this Use Case provides common everyday range capability, or a high-tech solution that “pushes the envelope” of what your range/TENA could do. Identify the key requirements here so there is no ambiguity about what is needed.

1.2.6 Object Model information. In this section of the template, please provide information on all of the objects, or class descriptions, that will be needed to support this Use Case. You do not need to fill in the attributes and methods for existing TENA class descriptions – just provide the class names as reference. If modifications are needed to an existing class description, use this template to identify the class and the attributes and/or methods that need to be modified. The first table provides guidance on identifying new or modified attributes. The second table provides guidance on identifying new or modified methods, or operations for this class. Additional lines, and/or copies of this sheet can be inserted into this template as required to handle additional class names, attributes, methods, and method parameters. A sample spreadsheet is at Figure [D-5](#).

1.2.7 Other relevant information. In this section of the template, please list all those critical aspects of the Use Case not captured elsewhere. A sample spreadsheet is at Figure [D-6](#).

APPENDIX D

EXAMPLE RADAR USE CASE

In a hypothetical example of development of an Object Model (OM) for a radar, the screens could look something like the figures shown in this Appendix. The following figures are shown as samples only. They are as follows:

- Figure D-1. Example radar Use Case (basic information).
- Figure D-2. Example radar Use Case (TENA applications).
- Figure D-3. Example radar Use Case (basic course of events).
- Figure D-4. Example radar Use Case (information flows).
- Figure D-5. Example radar Use Case (Object Model information).
- Figure D-6. Example radar Use Case (other relevant information).

	A	B	C	D
1				
2				
3	1.1 TENA Use Case Basic Information			
4				
5				
6	1.1.1 TENA User Name *	Kurt Lessmann	1.1.6 Background* This is an example OM submission for the DR&CG OM standardization process and should not be considered accurate nor representative. The OM submitted is only for example purposes. The radar OM is an example OM submission for a standard radar OM based on the TENA metamodel. This radar OM was designed to support the range community's requirements for both a simple radar system and a more complex, or extended, radar system	
7	1.1.2 Use Case Name *	TENA Radar OM		
8	1.1.3 Basic/Complex *	Basic		
9	1.1.4 Customer	TENA Community		
10				
11			1.1.7 Use Case Overview*: This use case describes a simple range test event with a range radar being controlled by a remote radar controller application with a 3d viewer displaying the events and a data logger recording the events	
12	1.1.5 Other Use Cases Referred to	None		
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24	1.1.6 Key Words	Radar, BasicRadarSystem, ExtendedRadarSystem, Antenna, Receiver, Transmitter, Logger		
25				
26				
27	starred (*) blocks are required			

Basic Info / TENA Applications / Events / Information Flows / Object Model / Other Info

Figure D-1. Example radar Use Case (basic information).

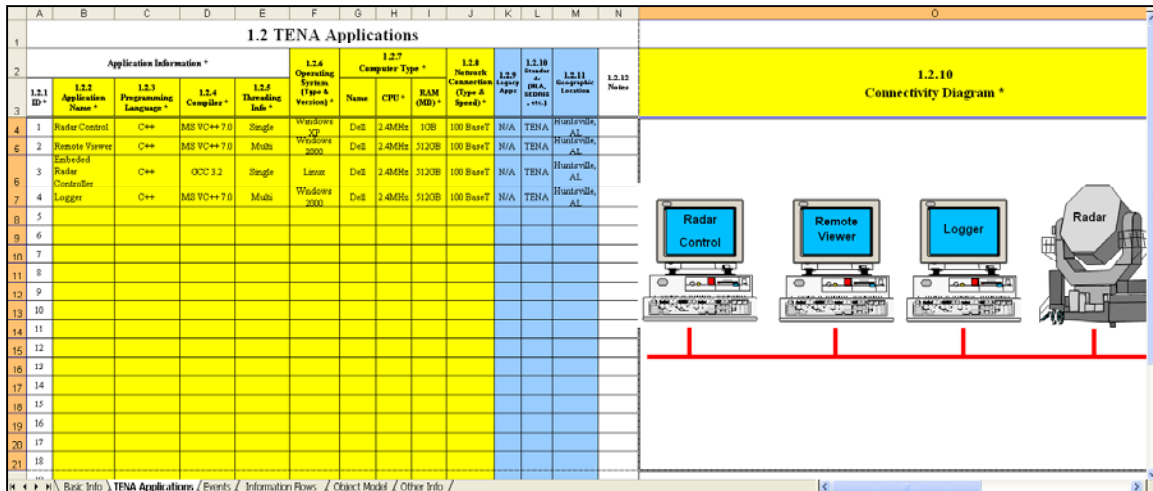


Figure D-2: Example radar Use Case (TENA applications).

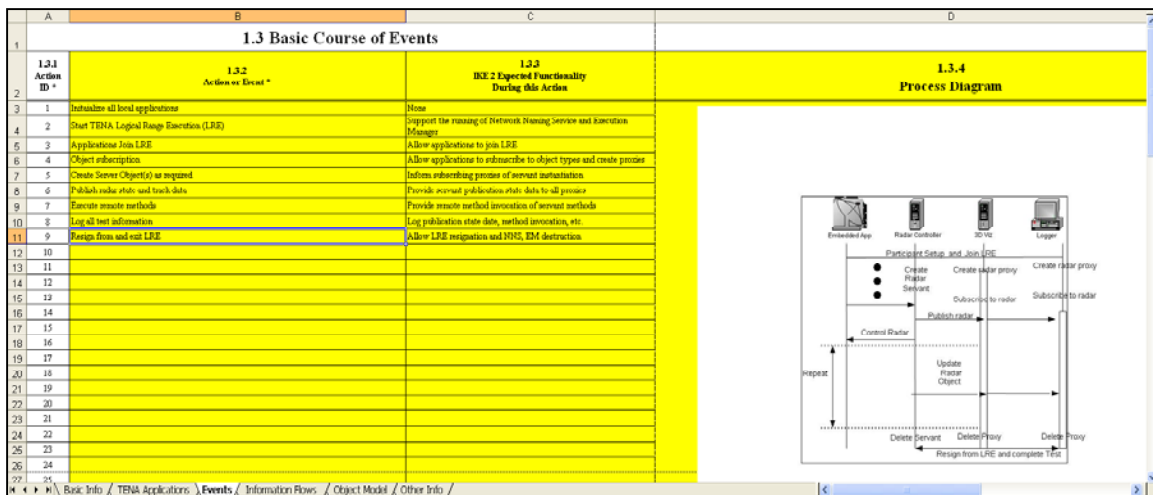


Figure D-3. Example radar Use Case (basic course of events).

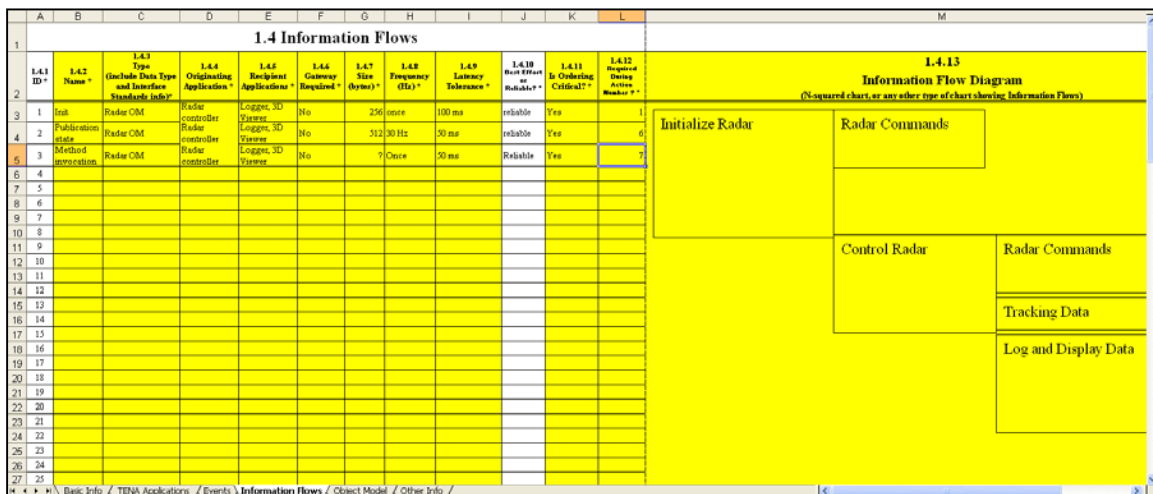


Figure D-4. Example radar Use Case (information flows).

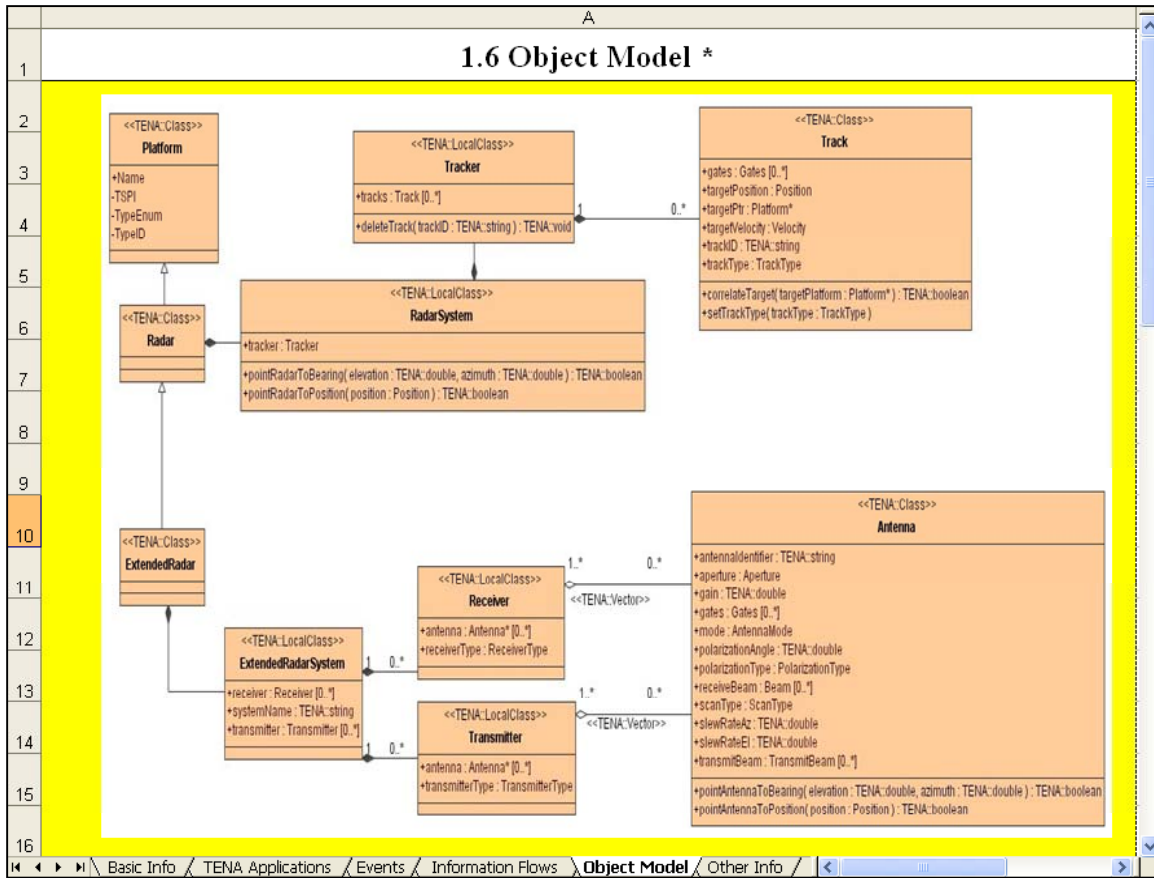


Figure D-5. Example radar Use Case (Object Model information).

A		B
1	1.7 Other Relevant Information as Required - Optional	
2	Note #	Note Comment
3	1	
4	2	
5	3	
6	4	
7	5	
8	6	
9	7	
	8	

Basic Info / TENA Applications / Events / Information Flows / Object Model / **Other Info**

Figure D-6. Example radar Use Case (other relevant information).

BIBLIOGRAPHY

Booch, Grady, and James Rumbaugh, Ivar Jacobson. (1999). The Unified Modeling Language User Guide. USA: Addison-Wesley.

Eriksson, Hans-Erik and David Fado, Magnus Penker, Brian Lyons. (2004). UML 2 Toolkit. Indianapolis, IN: Wiley Publishing.

Lafore, Robert. (1995). Object-Oriented Programming in C++ Second Edition. USA: Waite Group Press.

Weisfeld, Matt. (2000). The Object Oriented Thought Process. Indianapolis, IN: Sams Publishing.